COMMODORE USERS

Ahoy!

SPECIAL C-128 SECTION

48355 K

$2.50/ CAN. $3.00 SEPT. 1985

. . . MORE PROGRAMS THAN EVER BEFORE ! . . .

A LOOK INSIDE THE 1571
- SPECIFICATIONS
- ELECTRONICS
- MECHANICALS

FASTNEW
20-SECOND FORMATTER FOR THE 1541

CADET'S COLUMN
A NON-CRASH COURSE IN DISK MAINTENANCE

TOURING THE C-128 KEYBOARD

SUPER 64 UTILITIES!
- FILE LOCK
- START & END ADDRESS
- GO-LISTER
- AUTO-GEN

MANIPULATE SPRITES IN YOUR ORIGINAL GAMES— AS EASY AS **A PIG IN A POKE**

THE GREATEST GAMES!
- SUPERHERO
- MOXEY'S PORCH
- INVECTIVE
- FISH MATH

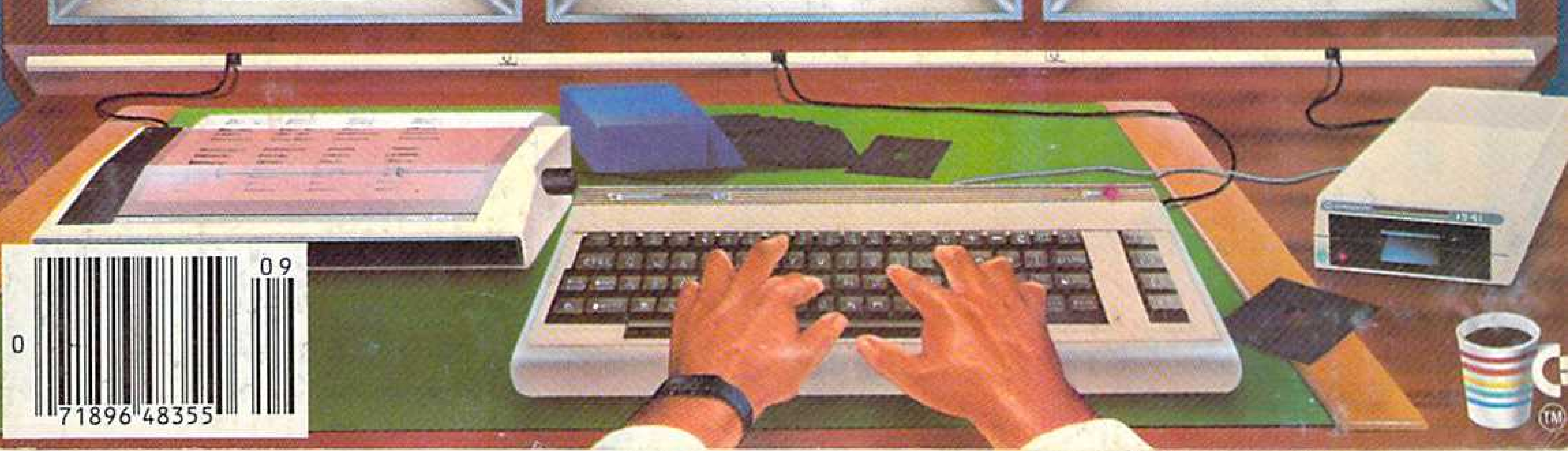
A NATURAL!
TUMBLING DICE AND DATA STRUCTURES
TAKING CHANCES ON YOUR VIC OR 64

AHOY! DOCK
MAKE YOUR COLLECTION SHIPSHAPE!

DRAGON TYPE
INPUT WITH BLAZING SPEED!

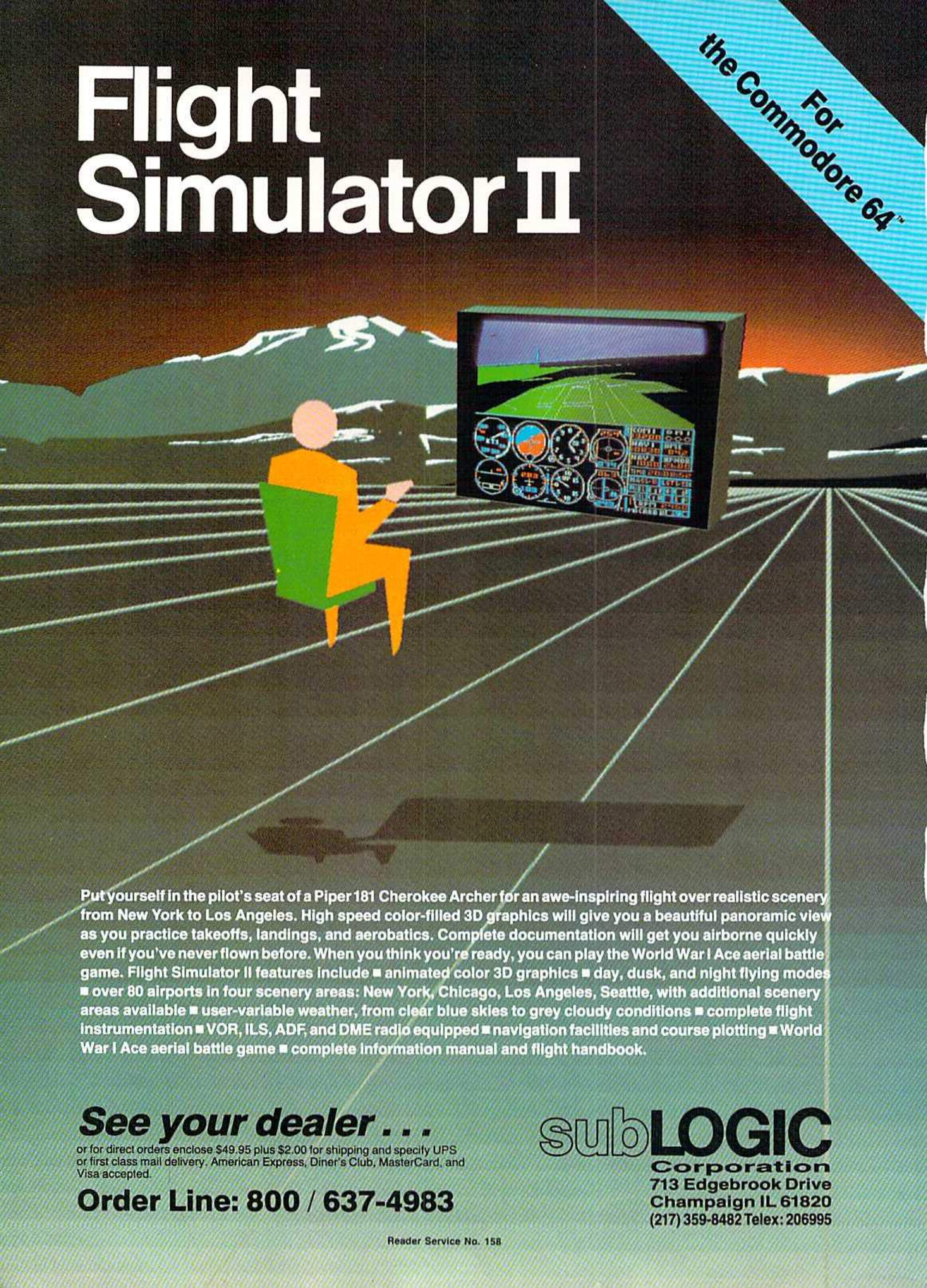
REVIEWS OF
- COMPUTEREYES
- MISTER ROGERS
- ON-FIELD FOOTBALL

AND MORE FOR THE VIC AND 64...END



09

0 71896 48355



For the Commodore 64™
Flight Simulator II
Put yourself in the pilot's seat of a Piper 181 Cherokee Archer for an awe-inspiring flight over realistic scenery from New York to Los Angeles. High speed color-filled 3D graphics will give you a beautiful panoramic view as you practice takeoffs, landings, and aerobatics. Complete documentation will get you airborne quickly even if you've never flown before. When you think you're ready, you can play the World War I Ace aerial battle game. Flight Simulator II features include ■ animated color 3D graphics ■ day, dusk, and night flying modes ■ over 80 airports in four scenery areas: New York, Chicago, Los Angeles, Seattle, with additional scenery areas available ■ user-variable weather, from clear blue skies to grey cloudy conditions ■ complete flight instrumentation ■ VOR, ILS, ADF, and DME radio equipped ■ navigation facilities and course plotting ■ World War I Ace aerial battle game ■ complete information manual and flight handbook.
See your dealer . . .
or for direct orders enclose $49.95 plus $2.00 for shipping and specify UPS or first class mail delivery. American Express, Diner's Club, MasterCard, and Visa accepted.
Order Line: 800 / 637-4983
subLOGIC
Corporation
713 Edgebrook Drive
Champaign IL 61820
(217) 359-8482 Telex: 206995
Reader Service No. 158

Ahoy!

DEPARTMENTS

FEATURES

*Includes programs: *Pig in a Poke* and *Basic Sprites* (for the 64)
**Includes programs: *Dice Simulator* and *Dice Analyzer* (for the VIC and 64)
***Includes programs: *The Quest* and *Response* (for the 64)

PROGRAMS

Cover art by James Regan; art and photos inside screens by Morton Kevelson, James Regan, and Javier Romero

ISSUE NO. 21 SEPTEMBER 1985

Ahoy! (#8750-4383) is published monthly by Ion International Inc., 45 W. 34th St., Suite 407, New York, NY, 10001. Subscription rate: 12 issues for $19.95, 24 issues for $37.95 (Canada $26.95 and $49.95 respectively). Second class postage paid at New York, NY 10001 and additional mailing offices.

Postmaster, send address changes to *Ahoy!*, 45 W. 34th Street, Suite 407, New York, NY 10001. Direct all address changes or matters concerning your subscription to *Ahoy!*, P.O. Box #341, Mt. Morris, IL 61054. All editorial inquiries and software and hardware to be reviewed should be sent to *Ahoy!*, 45 W. 34th St., Suite 407, New York, NY 10001.

VIEW FROM THE BRIDGE

"More Programs than Ever Before?" Haven't we used that pitch already? Yes—on our March '85 cover. That issue contained 8 programs, which at the time *was* more than ever before. This issue contains 14. Well—that's not entirely true. Our *Rupert Report, Creating Your Own Games,* and *Commodore Roots* columns include two programs each; so you could say 17. And if you count *Flankspeed* and *Bug Repellent...*

But let's not set the total *too* high. We don't want to make it impossible to top the *next* time we want to use the same cover blurb—which may be soon, as we add pages to your favorite Commodore magazine. (In fact, see next month's *View from the Bridge* for an announcement concerning an eight-page monthly addition to *Ahoy!*, covering the one major area of Commodore home computing that we've purposely neglected... until now!

In the meantime, our bursting-at-the-seams editorial policy makes it necessary to abridge our usual banter in order to describe the contents of this issue:

• Morton Kevelson continues his 128 opus with a detailed explanation of the new computer's keyboard and its 1571 disk drive. (Turn to page 47.)

• Orson Scott Card takes *Creating Your Own Games* into a territory you've long anticipated—sprites—and provides *Pig in a Poke*, a complete game for children. (Turn to page 18.)

• Dale Rupert continues his series of *Rupert Reports* on real-world simulations with this month's *Tumbling Dice and Data Structures.* (Turn to page 37.)

• Old timers as well as new will find useful information on disk maintenance and buying software in this month's edition of Cheryl Peterson's *Cadet's Column.* (Turn to page 81.)

• Mark Andrews concludes his two-part article on assembly language addressing in this month's *Commodore Roots.* (Turn to page 77.)

• *Fastnew* will cut disk formatting time from 90 seconds to 20—and, more important, will prevent your 1541's read-write head from banging during the operation. (Turn to page 33.)

• *Ahoy! Dock* will let you organize your collection of *Ahoy!* and other computer magazines for easy access to any article by category. (Turn to page 30.)

• *Start & End File Address* will generate the load and end addresses of any program. (Turn to page 17.)

• *Go-Lister* produces a printout of all BASIC program lines called by GOTO, GOSUB, THEN, and ON statements. (Turn to page 73.)

• *Dragon Type* is the latest programming pearl from *Ahoy!* ace Bob Spirko—this one, an entertaining typing tutor. (Turn to page 61.)

• *Superhero* generates a variety of mazes that you must negotiate in order to protect the world from falling meteors. (Turn to page 17.)

• You may find yourself hurling some *Invective* of your own as our game of the same name turns the tables on you every 15 seconds, scrambling the position of the joystick. (Turn to page 15.)

• *Fish Math* offers multiplication practice for young children, arcade action for adults, and, best of all, a reason not to convert your VIC 20 to scrap metal yet! (Turn to page 29.)

• *Auto-Gen* reads machine language routines from memory and turns them into data statements. (Turn to page 59.)

• *Moxey's Porch* requires speed and dexterity to zig-zag around a freshly painted porch. (Turn to page 73.)

• *File Lock* enables you to scratchproof any program file. (Turn to page 62.)

Of course, you'll find *Commodares* at their frustrating best (turn to page 55); a new collection of reader-generated computer graphics in our *Art Gallery* (turn to page 34); reviews of products like *On-Field Football, Blazing Paddles,* and the *Computereyes* video acquisition system (turn to page 63); and *Scuttlebutt,* featuring all the Commodore-related news from the Summer '85 Consumer Electronics Show (turn to page 7).

We remind you that our soon-to-debut programming tips column is awaiting your submissions (see page 46); our bulletin board system is ready to take your calls 24 hours a day (see page 114); and our programs, in this issue and every issue, are available on disk or cassette (see page 23).

It was our aim in producing this issue (as with every issue) to provide enough material to keep you stimulated, challenged, and occupied until next month. Even better, well into next month. We won't rest until we've got you hopelessly backed up on your reading.

—David Allikas

GUEST EDITORIAL

IMAGE

A Viewpoint by John C. Dvorak

Is the success of Commodore a fad? A fluke? A freak of nature? We'll all know in the next year or so. Whatever the case, it's quite likely that Commodore will have to fight an uphill battle to make a long-term success of the nifty Amiga Lorraine computer.

Commodore has created the image of a company that makes a cheap disposable computer. When articles are written about home computers sent to the closet to die, what computer do they talk about? There is always a picture of a Commodore 64 or VIC 20 languishing amongst the old ski boots. Hey, it isn't bad–it's disposable.

Image is everything in this business, as any IBM critic will tell you. The Commodore dilemma reminds me of two stories. The first goes back to the early days–the days of the SOL-20 computer. The SOL was released in 1977 and quickly found a following of users, many of whom attached a little Northstar floppy disk drive to the machine. A perfect combination, it was a compact system with the first commercial 5¼-inch disk drive. It was provided with a terrific BASIC and a nifty operating system. Above all it was simple and fairly inexpensive.

Processor technology, the company that made the SOL, wanted to do its own disk drive, so it spent a little over a year to come up with an expensive 8-inch hard sectored clunker called the Helios. This fiasco was probably as responsible as anything for grounding the whole Proc Tech business. The key to the whole blunder of the Helios is that if the company had seen what its customers wanted (cheap 5¼-inch disk systems) instead of what it thought it wanted (overpriced 8-inch clunkers) it would be in business today.

To Commodore marketeers this anecdote questions the likelihood of success if Commodore changes from a company that makes disposable computers to a company that makes hi-grade machines. Let's assume it can do it. There's still the image problem.

Anecdote #2. I once attended a focus group that showed a proposed Atari computer that was MS-DOS, CP/M, Apple, and Atari compatible. It would cost $1500. Nobody knew who the manufacturer was. The question was asked: "Would you buy this machine if it were an IBM?" It was one of those "highly likely, likely, unlikely, highly unlikely" type questions. Everyone said "highly likely." The same question was repeated, but this time the manufacturer was Apple. Again, the answer was "highly likely." The focus group jumped to Atari and the answer became "highly unlikely." The group saw the Atari as a game computer. This was a semi-serious machine–not an Atari.

Commodore was never even suggested at this group. Where would it stand? I shudder to think. Undoubtedly, it would be rated in the highly unlikely category. This will remain the case unless Commodore changes its image.

It must begin to improve its image now. Commodore isn't the first thing that comes to your mind when you think of the word "innovation." That lack is the first thing that has to go if the company is to survive. Words like "experimental," "insightful," "advanced," or even "well-made" do not bring to mind Commodore, either. What about words like "cheap computer" and "good deal"? Right! Commodore.

The Amiga will never be a truly "cheap computer" in the near term. It might be a relative good deal, but that's not what will sell the next generation of machines. Most users can get everything from a "good deal" to "trustworthy" and "reliable" from IBM. "Innovative" is the key word necessary for success.

Few companies have the wherewithal or opportunity to offer the public an advanced machine like the Amiga. Hopefully, the company will concentrate its marketing effort towards changing the public perception of Commodore. Hopefully it will create the right new image.

We sure don't want to see the Amiga in the closet with the ski boots, do we? ☐

Mr. Dvorak is the former editor of InfoWorld *magazine and writer of* Inside Track, InfoWorld*'s industry news column.*

WHITE HOUSE COMPUTER

P.O. BOX 4025, WILLIAMSPORT, PA 17701

In PA: 1-717-322-7700
1-800-351-3442

HOURS: Monday thru Friday 9:00 am till 6:00 pm

VISA and MC Accepted 4%

DISK DRIVES

C-64

1541	185.95
1571	249.95
1572	375.95
MSD Single	229.00
MSD Dual	459.00
Indus GT	239.00
Cardco SCD-1	199.00

INTERFACES

Tymac Connection/2K	62.95
Cardco +G	59.95
Cardco B	37.95
OR - D GPC w/2K	55.95
Grappler CD	89.95

PRINTER PAPER

Lazor 1000 shts	15.95
Lazor 2500 shts	29.95

DISKETTES ★

SKC

SS/SD	10.50
SS/DD	13.50
DS/DD	15.50

MAWELL

MD-1	15.95
MD-2	21.95

ELEPHANT

SS/SD	13.50
SS/DD	15.50
DS/DD	19.50

INNOVATIVE CONCEPTS

Flip-N-File

10	3.50
15	8.50
25	18.50
w/lock	
50	17.50
50	23.50
ROM Holder	17.50

KOALA

Pad Disk	39.95
Pad ROM	45.95

★ PRINTERS ★

C-64

MPS 801	179.95
802	208.95
803	175.95
MCS 801	314.95
DPS 1101	314.95

CITIZEN

MSP-10	325
MSP-15	479
MSP-20	469
MSP-25	629

EPSON

RX80	219
LX80	249
FX80	369
JX80	529
RX100	369
FX100	550

C. ITOH

8510 A	285
8510 BC1	329
8510 BC2	389
8510 BP1	339
8510 SP	389
8510 SR	429
8510 SCP	459
8510 SCR	477
1550 BCD	485
1550 P	445
A10-20P	465
F10-40P/S	875
F10-55 P/S	1059
F10 Tractor	169
F10 Sheet Feeder	349

MANNSEMAN TALLY

Spirit 80	249
MT 160L	495
MT 180L	595

STAR MICRONICS

SG10	214
SG15	379
SD10	345
SD15	450
SR10	485
SR15	585
Powertype	309

CARDCO

LQ 1	365
★LQ 3	209

PANASONIC

1090	187
1091	237
1092	389
1093	585
3151	455

LEGEND

880	209
1080	229
1380	269
1385	305

OKIDATA

182	219.95
84	640.95
92	349.95
93	560.95
Imagewriter	425.95

MONITORS

AMDEK

300G	119
300A	129
310A	145
Color 300	235
Color 500	345
Color 600	399
Color 700	469
Color 710	539

TEKNIKA

MJ-10	199
MJ-22	285

XTRON

Comcolor I	189

ZENITH

122	85.95
123	75.95

BMC

Amber	82.95
Green	79.95

COMMODORE

C-1702	199.95
C-1802	204.95
C-1902 RGB	259.95
C-1901 Monchrome	129.95

Dealers Inquiries Invited ★



SOFTWARE ★

BATTERIES INCLUDED

Paper Clip w/Spell Pk	69.95
Paper Clip	59.00
Home Pack	32.95
Bus Card II	124.95
B180 Board	99.95

BRODERBUND

Print Shop	29.95
Graphics Library	18.95
B.S. Writer	32.95

EPYX

FAST LOAD	23.95

VIDEO CASSETTE ★★ RECORDER $275.95

Symphonic VHS/Remote

- 105 Channel Cable Ready
- 14 Day / 1 Program Timer
- 12 Pre-set Channels
- Auto Rewind

Plus much, much more!

CHECK OUR PRICES!

COMPUTERS

C-64	149.95
C-128	275.95

PRINTER RIBBONS

DUST COVERS

MODEMS

C-1600	44.95
C-1650	54.95
C-1660	54.95
C-1670	185.95
300/1200 Band	

Cardco

MOD-1	59.95
C-64 MPP-1064	65.95
Tele Learning	49.95

Mitey Mo	59.95
Westridge	69.95
Compuserve Starter Kit	21.95

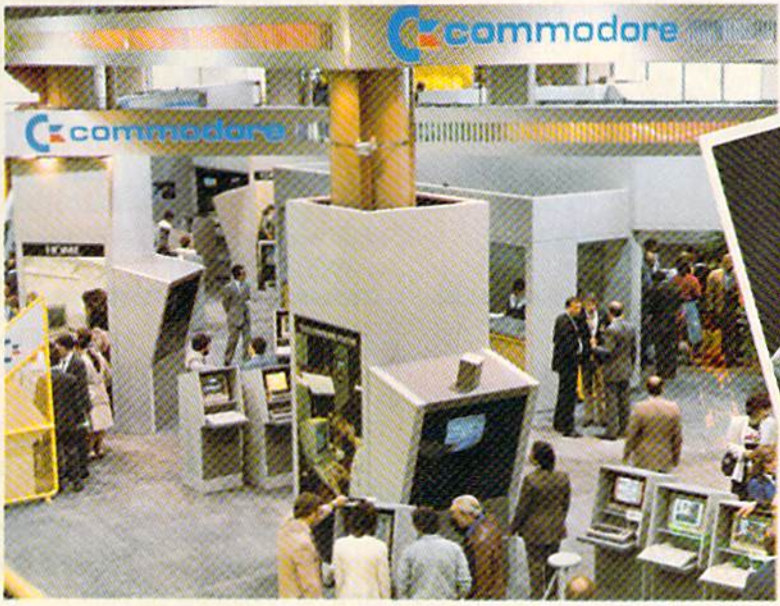
WE CARRY A FULL LINE OF: Batteries Inc., Broderbund, Contineal, Hess, Infocom, Innovative Concepts, Koala, LJK, Microbits, Micropose, Microtek, O.S.S., Professional S.W. Scarborough, S.S.I., Synapse, Time Works, Wico.

POLICY: No deposit on COD orders. FREE freight on all prepaid cash orders over $300 in the continental USA. APO and FPO add $5 per hundred. For priority Mail add $8 per hundred. PA Residents add 6% sales tax. Defective products must have Prior RA number. Schools net 15.

Reader Service No. 141

SCUTTLEBUTT

GAMES FOR C-64, C-128, AMIGA • COMMODORE DUAL DISK DRIVE & 1200 BAUD MODEM • 3D LIGHT PEN • 20MB HARD DRIVE • RAM/ROM EXPANDER CARTRIDGE • RGB MONITOR • MONEY MANAGER • ALPHACOM & COMMODORE PRINTERS • EDUCATIONAL RELEASES • TAX PROGRAM



MICHAEL DAVILA



Left: Commodore, star attraction of CES. Right: Commodore's PC10 and PC20 IBM clones. ***READER SERVICE NO. 104***

SUMMER '85 CONSUMER ELECTRONICS SHOW—THE CHIPS ARE DOWN

Still bullish on the home computer market? You weren't at June's Consumer Electronics Show. The *Wall Street Journal* aptly described the computer software section as a "ghost town." For us, the proof of the pudding was in the carrying: the bag of Commodore-related press releases we toted back to New York weighed in at twelve pounds, less than half of what we brought home from January's show in Las Vegas.

Most vendors were modest in their offerings, by comparison to the endless arrays of software displayed at last summer's show. Particularly noticeable was the dearth of educational releases for the Commodore, long the most glutted sector of the 64 market. A number of prominent manufacturers were conspicuous by their absence. Rumor has it that Atari, which had pulled out weeks before the show, was given their booth at a negligible cost to prevent the mass desertion the show's organizers feared their cancellation would signal.

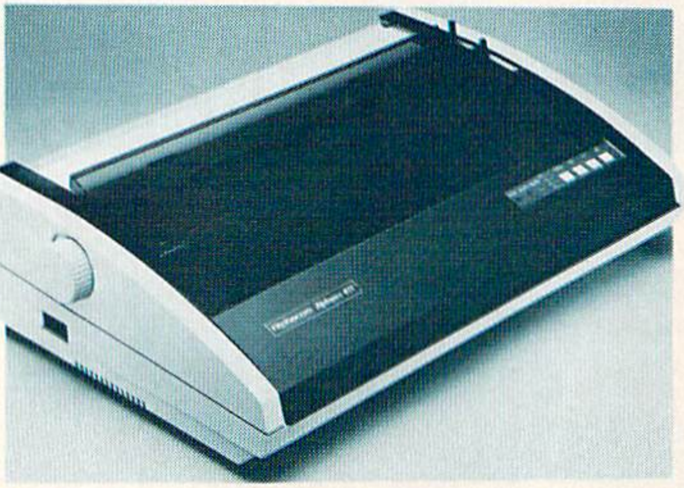
Of course, some believe that canny old Jack Tramiel cancelled primarily to convince Commodore not to rush the announcement of their new Amiga computer, since Atari's 520ST would not be there to steal the show in Amiga's absence. By reentering as an exhibitor at the eleventh hour—Jackintosh and all—Tramiel left Commodore with nowhere near sufficient time to ready an Amiga presentation. Believe this tale of corporate intrigue if you will. The fact is that, even without Amiga, Commodore dominated the floor (helped by the fact that Atari wound up tucked away in a corner of the upstairs level).

While the hordes of newshungry reporters that invaded the Commodore booth were disappointed by Amiga's absence, Commodore did display a most impressive collection of hardware—almost none of it, unfortunately, intended for sale in North America. Among the new computers planned for the European market are the IBM PC-compatible PC10 and PC20 (pictured on this page), the Unix-compatible Commodore 900 Business Computer (a multiuser workstation), and the 128D, featuring a built-in disk drive and independent keyboard.

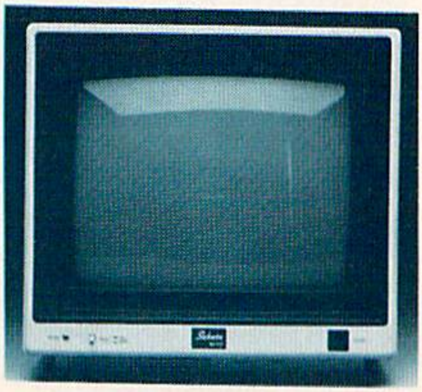
Of greater interest to American and Canadian readers were the following peripherals compatible with the C-128 (and in some cases the C-64 as well), some previously announced but seen for the first time at the convention:

• The 1572 Dual Disk Drive, comprising two horizontal 5¼" drives, each reading double-sided, single-density disks with up to 340K (formatted) memory in C-128 mode and 410K in CP/M mode.

• The Commodore 1670 Modem/1200, utilizing VLSI technology to



The compact (21 x 7 x 15.5") Alphapro 401 has a small desktop "footprint" and a sound cover that minimizes noise. Paper up to 15.7" wide can be used. READER SERVICE NO. 135



Sakata composite/RGB hi-res monitor. READER SERVICE NO. 136

deliver a 1200-baud rate at a price expected to be under $100. Using "AT" Hayes command protocol, the 1670 has auto-dialing, auto answer, auto speed, and auto mode selection.

• Compatible with the C-128 and C-64, the MPS 1000 Printer offers draft (100 cps), near letter quality (16 cps), and graphics modes, and a four-column display: Pica (80), Elite (96), Condensed Pica (132), and Condensed Elite (160).

• Just like the Macintosh's rodent, the Commodore Mouse will let 128 users sidestep special function keys and commands by moving the device over a desktop, thereby positioning onscreen icons.

Commodore also exhibited the software currently available for the 128: *Jane 2.0*, an integrated word processing/spreadsheet/filing program developed by Arktronics Corporation and permitting input with the Commodore mouse; the previously announced *Perfect* series from Thorn EMI (consisting of *Perfect Writer, Perfect Calc,* and *Perfect Filer*); and a 128 version of Commodore's own *Micro Illustrator.*

Doesn't sound like a lot? There's more: *Typing Professor* and *A Comprehensive Course in BASIC* will be coming from Commodore, as well as programs from such companies as Batteries Included, Timeworks, Scholastic, Spinnaker, Digital Solutions, Precision Software, and Melodian.

Still, an unimpressive amount of software for a computer announced five months ago—particularly a Commodore computer, the heir apparent to the 64 throne. Consider it one more indication of the depressed state of the home computer software business. Or as support for one last rumor—that a sizeable percentage of companies have already directed the bulk of their programming efforts toward producing software for the Amiga.

35CPS DAISY WHEEL

Aimed at the office market, the Alphapro 401 daisy wheel printer offers a print speed of 35 characters per second, path seeking logic, and a 2K (expandable to 16K) buffer for $599. The printer emulates the Diablo 630 printer protocols and uses Diablo- and Qume-compatible print wheels and ribbons.

Alphacom Inc., 2323 South Bascom Ave., Campbell, CA 95008 (phone: 408-559-8000).

RGB MONITOR

An alternative to the Commodore 1902, Sakata's SC-150 is a composite/RGB high resolution monitor with 13" screen, .42mm dot pitch, and dark face finish. It will operate with NTSC Composite, Y-C, RGBI, XRGB, and RGB input. Price $499.00.

Sakata U.S.A. Corporation, 651 Bonnie Lane, Elk Grove Village, IL 60007 (phone: 312-593-3211).

EDUCATIONAL RELEASES

Two C-64 programs for students in grades 7-12:

Homework Helper Math Word Problems ($32.95) aids students, through a series of questions and prompts, in building the correct equation to solve a problem.

Homework Helper Writing ($32.95) guides students through organizing and outlining their ideas for book reports, essays, and research papers, then writing the text with the program's word processing and printing capabilities.

Spinnaker Software, One Kendall Square, Cambridge, MA 02139 (phone: 617-494-1200).

Four classics-based C-64 programs from Fisher-Price Learning Software ($24.95 each):

Peter Rabbit Reading lets children 3-6 guide the famed bunny through a series of escapades, identifying consonant and vowel sounds and matching words to pictures.

Prokofiev's Peter and the Wolf Music introduces children 3-6 to the principles of music through seven games.

Jungle Book Reading teaches children 6-9 reading comprehension as they guide Mowgli the jungle boy through encounters with an assortment of animals.

The First Men in the Moon Math requires players 9-12 to help Professor Cavor solve the math problems with which the Sellenite creatures have blocked his way.

Fisher-Price Learning Software, P.O. Box 1327, Cambridge, MA 02238 (phone: 617-494-1222).

KIDWare has released 60 educational programs for the C-64, two to each tape ($9.95) or disk ($11.95), for children 1-16 years old. For more in-

formation contact KIDWare, P.O. Box 9762, Moscow, ID 83843 (phone: 208-882-3830).

Tales of Discovery makes the child the central character in two stories: *Pirates of the Soft Seas* (solve vocabulary and math puzzles to outwit pirates and find buried treasure) and *Fossils Alive!* (search for fossils and learn how a neighborhood looked in prehistoric times). For the C-64; $29.95.

Scholastic Software, 730 Broadway, New York, NY 10003 (phone: 212-505-3000).

From Sierra On-Line:

Goofy's Word Factory teaches grammer via three games involving conveyor belts and a kooky machine. For the C-64; $24.95.

Mickey's 3-in-1 ($49.95) consists of a word processor, data file manager, and telecommunications system designed for children. For the C-128.

Sierra On-Line, Inc., Coarsegold, CA 93614 (phone: 209-683-6858).

G'WAN, HAVE S'MORE

Cardco's *S'MORE* (for Super Memory Optimized RAM/ROM Expansion) is a $69.95 cartridge-based program that will provide your C-64 with 61,183 bytes of free RAM (that's including the 38,911 already present), over 60 new and enhanced BASIC commands and functions, full error trapping and automatic error help (which lists the error line and places a flashing cursor at the beginning of the program statement containing the error), full command selection, full up-down scrolling through program listings, and direct access to items normally PEEKed and POKEd.

Because *S'MORE*'s command structure is similar to the C-128's, C-64 programmers can make use of the advanced programming techniques available on the C-128, facilitating later conversion.

Cardco, Inc., 300 S. Topeka, Wichita, KS 67202 (phone: 316-267-3807).

SYNCALC TEMPLATES

For use with the *Syncalc* spreadsheet for the C-64, the *Syncalc Templates Disk* ($19.95) provides 22 commonly used forms and formulas for financial planning, budgeting, and record keeping. The disk also illustrates basic design principles for constructing or modifying special applications spreadsheets, from simple to complex.

Broderbund Software, 17 Paul Drive, San Rafael, CA 94903 (phone: 415-479-1170).

DEATH AND REVISIONS

Tax Command Planner, which is designed to let individuals test the tax effects of various financial alternatives, has been revised to account for Treasury Department proposals to eliminate many existing tax shelters. Purchasers of earlier versions can obtain the update by returning their program to the manufacturer. For new users, price of the 64-compatible version is $49.95.

Practical Programs, Inc., P.O. Box 93104, 625 N. Milwaukee St., Milwaukee, WI 53202 (phone: 414-278-0829).

ELECTRONIC COOKBOOK

The Electronic Cookbook lets C-64 users store, display, edit, and print recipes, 31 of which are included on the disk. Price is $20 (Canada $25) plus $2.50 postage.

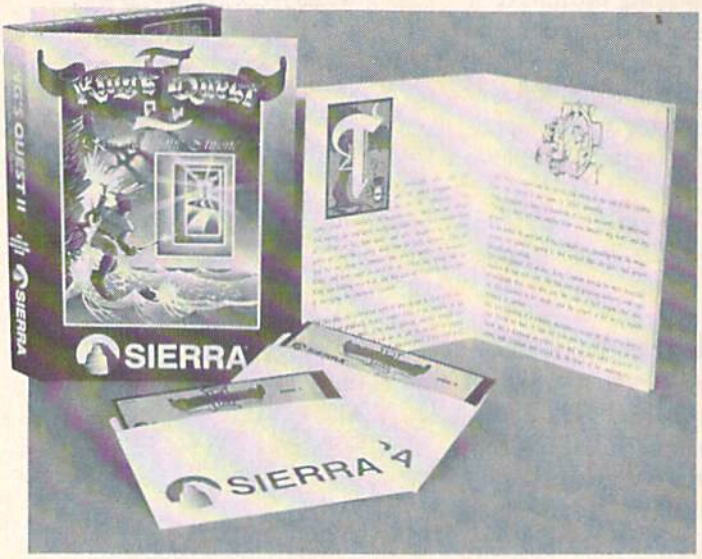
Arcturus Software, 2332 Palisade Dr. SW, Calgary, Alberta, T2V 3V1 Canada (phone: 403-281-1632).

KWIK-PITCH

Datamost has followed up the release of *Kwik-Load!* and *Kwik-Write!*, the first two releases in its *Kwik-Ware!* line of $19.95 software for the C-64, with *Kwik-Spell!* (spelling checker for *Kwik-Write!* and other C-64 word processors), *Kwik-File!* (database management), *Kwik-Calc!* (spreadsheet), *Kwik-Paint!* (graphics editor), *Kwik-Check!* (check balancing and maintenance), *Kwik-Pad!* (desk secretary), and *Kwik-Phone!* (communications).

Each *Kwik-Ware!* program has the *Kwik-Load!* fast loading program built in, and most are compatible with other programs in the series.

Datamost Inc., 19821 Nordhoff St., Northridge, CA 91324 (phone: 818-709-1202).



King's Quest II: Romancing the Throne *(from Sierra On-Line, Inc.) is to our knowledge the first game announced for Commodore's Amiga computer. A C-128 version will also be available. READER SERVICE NO. 137*

GAMES FOR THE 64, 128, AMIGA

Two graphics-and-text adventure games from Spinnaker's Telarium subsidiary:

Perry Mason: The Case of the Mandarin Murder takes place in the courtroom, where the player can do everything a courtroom lawyer might, from cross-examining and introducing evidence to consulting privately with the judge. The degree of characters' cooperativeness varies as a result of the player's interaction with them.

Based on the series of science fiction novels by Roger Zelazny, *Nine Princes in Amber* enmeshes the player in negotiation, politics, and alliances as, in the role of the prince Corwin, he battles his brothers and sisters for the throne of Amber. Again, the game's outcome depends heavily upon the player's interaction with the other characters.

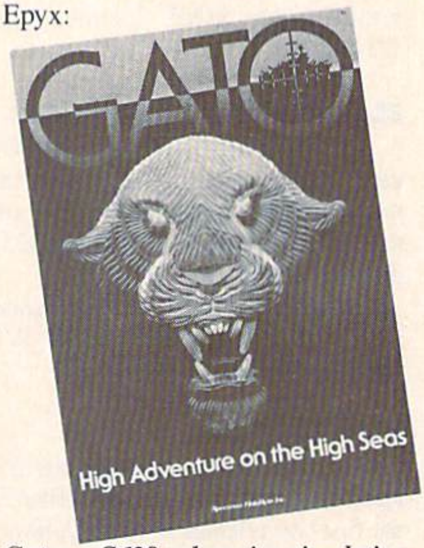
For the C-64; $32.95 each.

Telarium Corp., One Kendall Square, Cambridge, MA 02139 (phone: 617-494-1200).

Lode Runner's Rescue casts the player as Alexandra, daughter of the original Lode Runner who is being held prisoner deep underground by the evil Bungeling Empire. She must negotiate 46 mazes on her way to free her father. Of course, included is a game editor that lets the player construct an unlimited number of mazes full of elevators, trap doors, enemy guards, and rushing rivers. For the C-64; $29.95.

Synapse Software, 17 Paul Drive, San Rafael, CA 94903-2101 (phone: 415-479-1170).

Assorted new C-64 releases from Epyx:



Gato: *a C-128 submarine simulation. READER SERVICE NO. 138*

Winter Games features six sports, including ski jumping, a ski biathlon and hotdogging, speed and freestyle events, and a bobsled race.

The World's Greatest Football Game allows coaches to develop their own plays or customize existing ones, and store up to 120 of them in a playbook. The action can be viewed from a sideline, bird's-eye, or closeup view.

Joining *Ballblazer* and *Rescue on Fractalus!*, previous releases in Epyx's Lucasfilm Games Division: *The Eidolon,* centering around a 19th century machine that taps psycho-kinetic forces, transporting the player to a magic realm of caverns populated with Trolls, Greps, and Guardian Dragons; and *Koronis Rift,* compelling the player to wrest the secrets of the ancients from a genetically engineered race of guardians that outlived their creators.

Jet Combat Simulator recreates actual flight situations that confront an F-15 pilot, including cross-wind turbulence and blind landings. Four options let the player progress from landing practice to air-to-air combat.

The components of the *Temple of Apshai* trilogy–*Temple of Apshai, Curse of Ra,* and *Upper Reaches of Apshai*–will be sold together for the price of a single game.

All of the above Epyx games will retail for $29-$35 (except *Jet Combat Simulator* and the *Apshai* trilogy–$19-$29).

Epyx, Inc., 1043 Kiel Court, Sunnyvale, CA 94089 (phone: 408-745-0700).

New for the 64 from Activision:

As *Hacker* begins, the player has broken into an unknown computer system that conceals a mystery. With no background information, no rules, and no clues, he must find out whose system he's hooked into and what's going on.

Fast Tracks: The Computer Slot Car Construction Kit lets the user build a race course piece by piece from a variety of types of track sections, view the course as a whole, and then race his own custom-designed car. Favorite courses can be saved to disk for future use.

Though certain to bring a sneer to Orson Scott Card's lips, Garry Kitchen's *Gamemaker: The Computer Game Design Kit* joins the ranks of programs that make it possible to create games without programming knowledge. The user selects characters (rocket ship, running man, etc.) from a menu, along with their speed and direction, background, sound effects, and music.

Activision, Inc., 2350 Bayshore Frontage Road, Mountain View, CA 94043 (phone: 415-960-0410).

Three new C-64 simulations from MicroProse, each $34.95:

Acrojet, The Advanced Flight Simulator, for computer pilots who've won their wings in *Solo Flight*, offers a Decathlon of Sport Aviation events, including spot landings, ribbon cuts, acrobatics, and other high performance maneuvers.

Gunship, The Helicopter Simulator, simulates operation of the new AH-64 Apache attack helicopter and features multiple weapon and navigation systems.

Silent Service, The Submarine Simulation, provides a selection of historical scenarios from single ship attacks to multipatrol missions with graduated enemy skills.

MicroProse Software, 120 Lakefront Drive, Hunt Valley, MD 21030 (phone: 301-667-1151).

C-64, C-128, and Amiga releases from Sierra On-Line:

Gato, a submarine simulation previously released for IBM and Apple, will be converted to the C-128. The game transforms the player into a World War II sub commander, prowling the Pacific in search of the Japanese Imperial Fleet. Price: $39.95.

King's Quest II: Romancing The Throne, featuring 3D animated graphics, will be available in C-128 and Amiga versions. Price: $49.95.

Stunt Flyer ($39.95) will let C-64 users pilot a Pitts Special through a number of aerobatic sequences.

The Black Onyx ($39.95) takes C-64 adventurers down into a dungeon maze, then to the top of a mysterious tower.

Sierra On-Line, Inc., Coarsegold, CA 93614 (phone: 209-683-6858).

COMPUTER RACK SYSTEM

Designed for Commodore systems, the Hi-Rise computer/printer stand ($39.95) will allow users to store their C-64 or VIC 20 monitor, printer, and disk drive (or printer paper) in less than two square feet of desktop space. Crafted of solid oak, it's a sure bet to outlive the rest of your configuration.

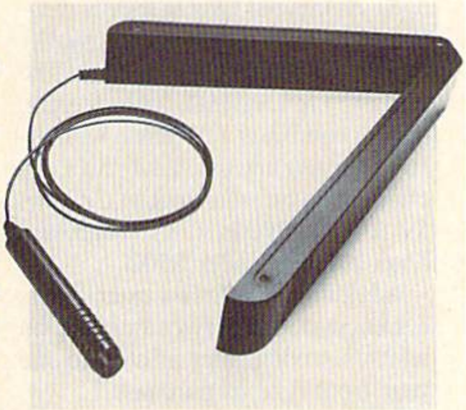
Unique Wood Products, P.O. Box 52, Mankato, MN 56001 (phone: 507-625-5123).

BIC ROGERS?

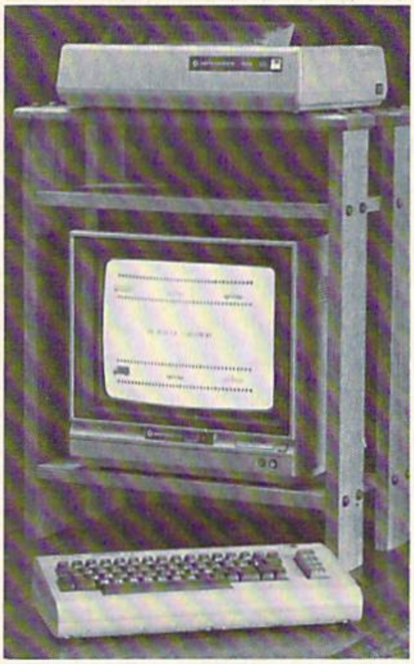
The Space Pen input device differs from other light pens in its ability to detect the third dimension, or "Z" axis, enabling the user to operate in a six foot area in front of the computer display. While no software currently available can utilize this capability, the pen will work in two dimensions with all touch tablet paint programs and an assortment of popular packages.

C-64 or VIC 20 version $150, Amiga version $175, including sampler software.

Soniture, Inc., 480 Vandell Way, Campbell, CA 95131 (phone: 408-866-4616).



Space Pen detects third, or Z, axis.
READER SERVICE NO. 139



Hi-Rise stand: sturdy and practical.
READER SERVICE NO. 140

10-20 MEG HARD DRIVE

If you can live with the irony of spending $100 for your computer and 15-30 times that for a storage device, Fiscal Information has the peripheral to fulfill your most extravagant fantasies. Lt. Kernal, a hard disk drive for the C-64, will access data up to 43 times faster than the 1541 (though compatible with it in most applications). The drive also provides 34 additional or enhanced system commands, autoboot of a selected program, built-in CP/M-like command features, up to 10 logical drives, and the ability to keep up to seven files open simultaneously. Mass and archival backup and restore facilities are available.

10MB version $1595; 20MB, $1995. Streaming tape option, which backs up 10 megabytes in 2 minutes, if $1525.

Fiscal Information Inc., 143 Executive Circle, Daytona Beach, FL 32014 (phone: 1-800-874-1890).

TELECOM NEWS

General Electric is offering two hours of access to CompuServe's executive information service with the purchase of a $99.95 GE telephone modem, model 3-8200 (see May '85 *Scuttlebutt,* page 9).

General Electric Company, Electronics Park, Syracuse, NY 13221 (phone: 315-456-2446).

BRKTHRU gives the user access to over 65 databases covering business and finance, science, medicine, education, general reference, and more. Cost is $75 to sign up ($25 credit) plus usage charges ranging from $17.50-$125 an hour.

BRS Information Technologies, 1200 Rt. 7, Latham, NY 12110 (phone: 518-783-1161).

SCREENPLAY REBATES

Software Resource Group will offer a $5 rebate direct to the consumer on purchase of all Screenplay and Brown Bag Software.

Software Resource Group, 1095 Airport Road, Minden, NV 89423 (phone: 702-782-9731).

PRINT SHOP GRAPHICS

The Graphics Library, Disk Two, like the previously released *Disk One,* provides additional designs, symbols, and pictures for use with *The Print Shop,* this time in the categories of Jobs, Hobbies, People, Places, Travel, and Health. For the 64; $24.95.

Broderbund Software, 17 Paul Drive, San Rafael, CA 94903-2101 (phone: 415-479-1170).

MAKE YOUR OWN KIND OF MUSIC

The two latest additions to MasterSoft's Mastery in Music series, *Trumpet Master* and *Clarinet Master,* let the student personalize his practice music with his choice of key and time signatures, note type, and note range. Instant access to music facts and fingerings is always available. Scales, thirds, and intervals in every major key are included. For the C-64; $49.95 each.

MasterSoft, P.O. Box 1027, Bend, OR 97709 (phone: 503-388-7654).

A FIRST FROM JUKI

Juki's first dot matrix printer, the 5510, churns out type at speeds ranging from 30 cps (near-letter quality mode) to 180 cps (draft mode). The printer's bidirectional, logic-seeking system combines a 9" platen with assorted printing width capabilities. Included is a built-in tractor, 3K buffer (expandable to 15K). Suggested retail price is under $500.

Additionally, Juki has extended its current customer service warranty on all its microcomputer printers to one year from date of purchase.

Juki Office Machine Corp., 299 Market St., Saddle Brook, NJ 07662 (phone: 800-932-0590).

MORSE CODE TUTOR

IMCT offers a tutorial in International Morse Code, and a training program to increase your code speed from 1-25 words a minute. For the 64; $18.95 (PA residents add 6% sales tax).

AC3L Software, P.O. Box 7, New Derry, PA 15671.

ACTIVISION RELEASES

Two non-games for the C-64 from Activision:

The Complete Computer Fireworks Celebration Kit lets patriots create fireworks displays, add music and special messages, and turn their displays into computer greeting cards.

Alter Ego allows users to vicariously experience life as someone else, building a unique personality through reactions to a variety of emotional, physical, and other situations.

Activision, Inc., 2350 Bayshore Frontage Road, Mountain View, CA 94043 (phone: 415-960-0410).

BOOKS

The Greatest Games: The 93 Best Computer Games of All Time (by *Ahoy!* reviewers Dan Gutman and Shay Addams) reviews 93 games in 18 categories. Price is $9.95.

COMPUTE! Publications, Inc., 324 W. Wendover Ave., Suite 200, Greensboro, NC 27408 (phone: 919-275-9809).

The latest edition of the *Softsel Product Encyclopedia,* covering more than 3,000 software and hardware products in its 334 pages, is available for $29.95.

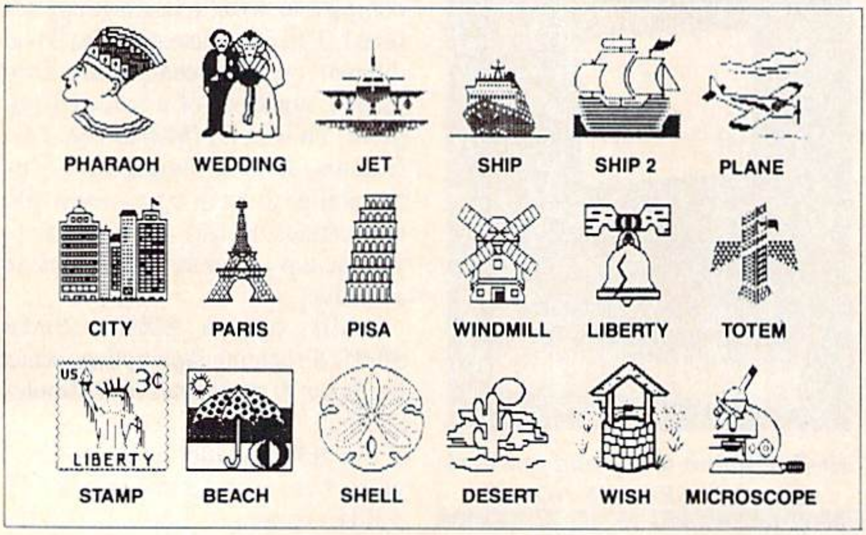
Softsel Marketing Dept., P.O. Box 6080, Inglewood, CA 90312 (phone: 714-640-7375).

Problem Solving in BASIC with the Commodore 64 consists of a book and a 30-minute videotape that take the user on a tour of top-down programming on the 64. Price is $59.95.

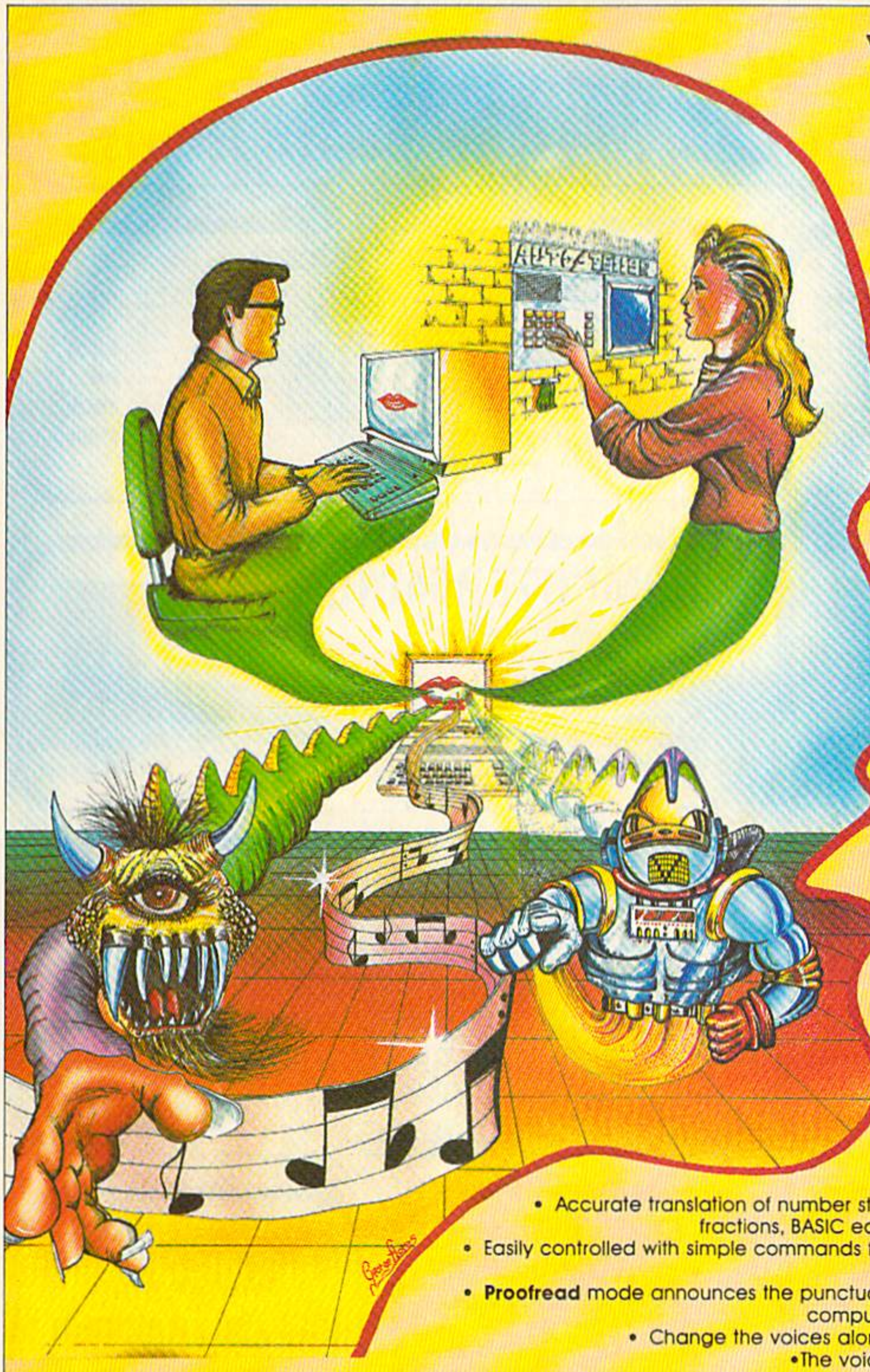
Hayden Book Company, 10 Mulholland Drive, Hasbrouck Heights, NJ 07604 (phone: 201-393-6306).

Marketing Your Software ($16.95) offers a 26-step plan, from defining objectives and analyzing competition to investigating distribution and designing packaging.

Addison-Wesey, General Publishing Group, Reading, MA 01867.



Graphics Library: ***pix for Print Shop.***
READER SERVICE NO. 150



YOUR MISSION: TO EXPLORE NEW WORLDS

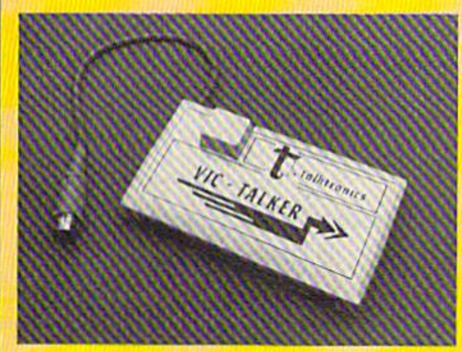
Tap into an exciting computer resource available for the VIC-20 and C64 computers: synthesized speech.

Talktronics brings to you two extremely powerful plug-in speech synthesizers to let your computer sound off in ways you never knew were possible.

VIC-Talker and **64-Talker** enhance your educational programs and games, aid the handicapped and generally increase computer "friendliness".

Look at these sophisticated features:

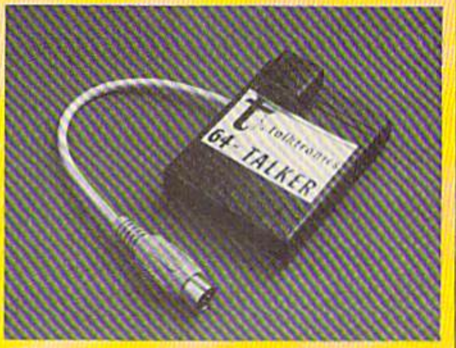
- Self contained firmware – disk or tape not required for synthesis.
- Audio mixer to combine computer generated sounds with the synthesized voice for **VIC-Talker** or **64-Talker.**
- Two user-programmable voices that can be as varied as your imagination – from chipmunks to robot warriors.
- Translation of ordinary text directly to synthesized speech using pronunciation rules for English.
- Nonvolatile user-programmable exception word memory permits special pronunciation or translation of words you select and is retained when power is off.
- Contains rules for pronunciation of text, numbers, punctuation and BASIC command words.
- Accurate translation of number strings in decimal, dollars and cents, fractions, BASIC equations, time and phone numbers.
- Easily controlled with simple commands from BASIC or machine language – like using the printer or disk.
- **Proofread** mode announces the punctuation in a different voice – lets your computer "speak" BASIC program listings.
- Change the voices along with speaking text. Make it sing!
- The voice comes out on the TV or monitor.



VIC-Talker and 64-Talker
are priced at $89 each
and are available from

Talktronics, Inc.
27341 Eastridge Drive
El Toro, CA 92630

(714) 768-4220
9 AM - 5 PM, PST



Sensational Prices!
. . . On Our Most Popular Items!

Introducing . . .

THE 99¢ DISKETTE

Are you paying too much for diskettes? Try our first quality, prime, 5¼" diskettes (no rejects, no seconds) at these fantasic sale prices and save, save, SAVE! Disks are packaged in boxes of 50; each box contains 5 shrink-wrapped 10-packs that include diskettes in sleeves, labels, and write-protect tabs.

Each diskette is certified to be 100% error free and comes with a lifetime warranty (if you have a problem, we'll replace the diskette). All diskettes include hub reinforcement rings and write-protect notch.

All diskettes are double density and work in either single or double density drives.

SS, DD Diskettes, Box of 50
32391 — $49.50 – $99¢ ea.!

DS, DD Diskettes, Box of 50
32403 — $64.50 – $1.29 ea.!

LOWEST PRICES!

We can offer you some of the lowest prices in the country on the most popular printers, monitors and interfaces. Our normal prices are already low, but to make sure you get the best deal you can, we will also meet most competitive prices in this publication when placed on an equal basis (Remember—we don't charge for use of your credit card, impose excessive shipping fees, or use any other hidden extras to boost the price you pay. Due to the rapid change in prices in the computer industry, we can only meet prices at the time you place your order; we cannot adjust prices on items ordered or shipped on an earlier date.) Another plus for charge card customers—your charge card is billed at time of shipment only for the items shipped—no early billing, no long wait for the merchandise you already paid for.

star · EPSON · BMC · AXIOM · CARDCO

★ THE BEST PRICES ★ THE BEST SERVICE ★
WHY SHOP ANYWHERE ELSE?

COMPUTERFACTS™ from SAMS

Technical Service Data for Your Computer and Peripherals



Sams introduces COMPUTERFACTS™ . . . packets of information that reveal the inner workings of major brand micros, monitors, printers and disk drives. Includes schematic wiring diagrams, parts lists, disassembly instructions, troubleshooting techniques, and other repair data.

- 33477 Commodore 64 Computer
- 33513 VIC-20 Computer
- 33551 Commodore Plus/4 Computer
- 33481 Commodore 1541 Disk Drive
- 33496 Commodore 1525 Printer
- 33508 Commodore C16 Computer
- 33546 Commodore 1701 Monitor

All packets $17.95 each

The Most Complete Disk Drive You Can Buy

Includes Word Processor, Spread Sheet, and Data Base Manager

INDUS GT ONLY $259

State-of-the-art disk drive for C-64 and VIC-20 features LED sector indicator, electronic write-protect, built-in wedge, and operating speeds up to 400% faster. Package includes cables and power supply, carrying case that doubles as a disk storage file AND software starter package featuring word processor, spreadsheet, and database manager program!
FULL YEAR PARTS AND LABOR WARRANTY!!

33443 Indus GT Disk Drive — ONLY $259

GRAPHICS PRINTER INTERFACE



The PPI (Parallel Printer Interface) from DSI is a high-performance graphics printer interface for your Commodore 64 or VIC-20, offered at an unbelievable price! The PPI fully emulates a Commodore printer so that all software written for Commodore computers and printers can be run with the PPI and most any parallel printer. Print true Commodore graphics and alpha-numerics as well as any special character sets offered by your specific printer. Works with Star, Epson, BMC, and many more. Comes complete with all necessary cables for quick installation and includes easy-to-follow user's manual.

33565 Parallel Printer Interface — **$44.95**

CARTRIDGE EXPANDER

NAVARONE

Plus $22.95 FAST LOAD OFFER!

Stop wearing out your computer and cartridges by endless swapping! New design cartridge expander features two cartridge slots. Select either cartridge, or the "off" position if no cartridge is to be used. Reset the computer independently of the power switch. Cartridge slots are vertical for easy access—no blind fumbling behind the computer.

34220 2-Slot Cartridge Expander — $19.95

The cartridge expander is a great companion for the Epyx Fast Load cartridge—you can keep it permanently installed plus have a slot free for other cartridges!

34216 Fast Load Cartridge (Sug. Retail $34.95) — $29.95

Fast Load ONLY $22.95 with purchase of cartridge expander!

DUST COVER and "EVERYTHING BOOK" INTRODUCTORY SPECIAL

Get to know us by ordering this great dust cover for your VIC-20 or Commodore 64 and our catalog, "The Everything Book for the Commodore C-64 and VIC-20 Home Computers," for $2.95 (no extra shipping or handling charges). Cover is antistatic, translucent, 8 gauge vinyl sewn to our exacting standards with reinforced seams. Discover the savings and easy shopping available from Tenex Computer Express!



$2.95

31627 Dust Cover and Catalog

USE SINGLE SHEETS IN YOUR TRACTOR FEED PRINTER WITH

PAPER TRACTOR®

Paper Tractor carries ordinary sheet paper through your tractor-feed printer!
Simply place your letterhead, checks, invoices, forms, even legal-size sheets into the durable flexible plastic carrier and feed into your adjustable tractor-drive printer. Works with any printer, carries any paper.

22084 — **$11.95**

From Your Friends At TENEX Computer Express™

We gladly accept mail orders!

P.O. Box 6578
South Bend, IN 46660

Questions? Call 219/259-7051

Ad A1A

SHIPPING CHARGES

ORDER AMOUNT	CHARGE
less than $20.00	$3.75
$20.00–$39.99	4.75
$40.00–$74.99	5.75
$75.00–$149.99	6.75
$150.00–$299.99	7.75
$300 & up	8.75

MasterCard · American Express · VISA

NO EXTRA FEE FOR CHARGES

NO HIDDEN EXTRAS

ORDER TOLL FREE 1-800-348-2778

INVECTIVE

for the C-64

By George Trepal

Invective isn't like any other game you've ever played. Nothing chases you and you're not shooting at anything. You simply move around the game board collecting points by snapping up green figures at 10 points each.

Of course there's a catch. The joystick is scrambled so that what used to move up now moves perhaps left, what used to move left now moves perhaps right, etc. The joystick is scrambled every 15 seconds to a new randomly chosen configuration. If you bump a blue figure the joystick gets rescrambled and you lose 20 points. If you bump a yellow figure you lose 50 points.

In the first 15 seconds of the 90 second game, the joystick operates normally. After that it's hard going all the way, since everything is against you. You'll be lucky to make a score as high as zero the first few times you play *Invective.*

That's all you need to know to play. From here on I'll just ramble about technical matters.

When you type *Invective* there is no need to copy the blank lines or REM statement lines.

Line 430 tells the computer to scramble the joystick every 15 seconds. If you'd like a different number of seconds, substitute your number for 15 at both places in the line. Line 680 limits play time to 90 seconds, which is expressed as 130 (for 1 minute 30 seconds) in the line.

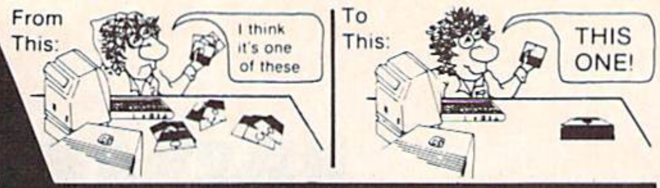
Lines 750 to 850 tell you how the joystick has been scrambled. The game is much harder if you don't have this information, so you might want to leave it out. The easiest way to do this is to change the command to print brown letters to a command to print medium gray letters. Since the background is medium gray the letters will be invisible. The print brown (Commodore key and 2 key) is in line 750 and is easy to change to print medium gray (Commodore key and 5 key).

You may be wondering how the joystick can be scrambled. The normal way to use the joystick is to first get the value in the joystick address (location 56320). This value is then ANDed with 15 to get rid of the meaningless first four bits of the byte. The result is then subtracted from 15. A result of 0 means no change, 1 means up, 2 means down, 4 means left, and 8 means right. So far that's all by the book and all that remains is IF statements. If the value is 1 move up, if 2 move down, and so on. But what would happen if we told the computer to move right instead of up when the value was 1? Answer: it would work just fine.

The values on how to move are held in the J( ) array and the pointers relating the J( ) values to the joystick are in the SA( ) array.

Lines 690 and 700 shuffle the SA( ) array. The algorithm starts with the first element of an array and exchanges it with a randomly chosen element of the array. It then repeats the process with the second, third, etc. elements. This technique is useful for computer card shuffling. ☐ *SEE PROGRAM LISTING ON PAGE 112*

C-64 and C-128* OWNERS:



Now YOU too can get the "Software Edge!"

Do you know the full contents of all your Disks in your Software Library?

Stop playing "Hide-and-Seek" with your Disk software! Let your Commodore 64™ "memory" do all the work for you. Easy Manual and Program Menu-Screens make everyone a "pro" in 1 day!

Here's a small "PEEK" at SOME of the FULLY-INTEGRATED POWER of PRODUCTIVITY at your fingertips. . .

MASTERDISK©:

- a SORTED Master Catalog of ALL your Disks (even those un-listable "protected" Directories: Names, ID's, File info, etc.)
- Search + List any File/Disk/ID in seconds, automatically, in any of SEVEN WAYS.
- Transfer ML-Programs or Files from Disk to Disk (CATEGORIZE your software)
- Stop disaster before it strikes: change duplicate ID's fast, without total re-copy.
- Restore accidentally-scratched files in a flash, with VERIFICATION that they're OK.
- Works with 1 or 2 Single Disk Drives (1541, MSD, etc.) with or without Printer (Commodore or ASCII).
- Handy-Reference Card for HEX/ASCII/ BINARY-TO-DECIMAL VALUE TRANSLATION + FORMULAS, included FREE.
- All this + MUCH MORE, for just $34.95. (U.S.A.), $45.95 (Canada)

NEW!! - MASTERDUAL©, "Son of MASTERDISK©" . . . same power, but with Dual Drive Speed!

- Same Catalog Features as MASTERDISK©
- Same Disk repair functions, too
- For all C-64 & C-128* systems with Dual Disk Drives (4040, MSD, etc.)
- Utilizes all Dual Drive functions for maximum speed and ease of use
- FREE Handy Reference Card for HEX/ASCII/ Binary to Decimal Translation + Formulas
- Experience FULLY-INTEGRATED POWER for just $29.95 (U.S.A.), $39.95 (Canada)

***MASTERDISK©** and **MASTERDUAL©** are also compatible with the NEW C-128 (in C-64 Mode)!

Don't MESS with anything LESS! ***OUR CUSTOMERS WRITE:***

"Been meaning to write and let you know how much I've enjoyed it . . . So, let me say that MASTERDISK is EXCELLENT." — Middletown, NY

". . . just the thing I need." — APO, NY (Germany)

"Hope to get my [MASTERDISK] yesterday." — Maple Grove, MN

"I am impressed with your product and would like to be able to endorse it for TPUG." — Ontario, Canada

". . . nothing else even comes close!" — Burlington, IA

We ship within 24 hours! Send Money Order or Check to:

INTEGRATED-SOFTWARE SYSTEMS
P.O. Box 1801, Ames, IA 50010
VISA/M.C./C.O.D., CALL **(515) 233-2992**
9 am - 9 pm (Central Time) Mon.-Fri.
Add $3 Shipping/plus $2 Foreign, C.O.D.
IA Residents add 4% Sales Tax.

IN CANADA: Cricket Distribution Co.,
P.O. Box 953, Cobourg, Ontario, Canada K9A 4W4
VISA/M.C./C.O.D., CALL **(416) 372-3692**
9 am - 6 pm (Eastern Time) Mon.-Fri.
Add $3 Shipping/plus $2 C.O.D.
Ontario Residents add 7% Sales Tax

Whether you have tens or hundreds of disks, now YOU can be the MASTER of your TOTAL Software Library!

SUPERHERO

for the C-64

By James C. Hilty

A meteor storm has caused the space shuttle to move off course and rapidly gain altitude. Upon reaching an altitude of 800 miles, the shuttle will be lost forever. The only hope of saving it is you, the *Superhero.* You must destroy 10 golden meteors before the shuttle reaches 800 miles. But beware—hitting the green Zyptonite meteor will render you powerless and the game will end.

The game opens with the title screen, then a graphics chart depicting the Superhero, the space shuttle, and the meteors. The playing screen incorporates a randomly drawn maze. The Superhero character is at the upper right of the maze, while a Zyptonite meteor is at the upper left. A joystick in Port 1 allows you to move Superhero while he tries to get the 10 gold meteors which are placed along the maze. The Zyptonite meteor moves randomly throughout the maze. Colliding with the Zyptonite meteor finishes you off.

The altitude is constantly displayed along with the score. If the shuttle reaches 800 miles, the game is over. Each meteor you destroy is worth 10 points. Getting all 10 of the meteors before the shuttle reaches 800 miles gives you bonus points, lowers the altitude, and allows you to progress to a new maze.

The running Superhero figure and the space shuttle in the maze cutouts are sprites which react to the different conditions going on in the maze.

The main loop (lines 870-950) uses a machine language joystick routine to move the Superhero character. The Superhero in the maze cutout is animated by POKEing different values into SPRITE 0's memory pointer. A replay option is incorporated at the end of the game. ☐

SEE PROGRAM LISTING ON PAGE 101

START & END FILE ADDRESS

for the C-64

By David S. Smart

As well as giving the start or load address of a file, this little program will give the length of a file and the end address.

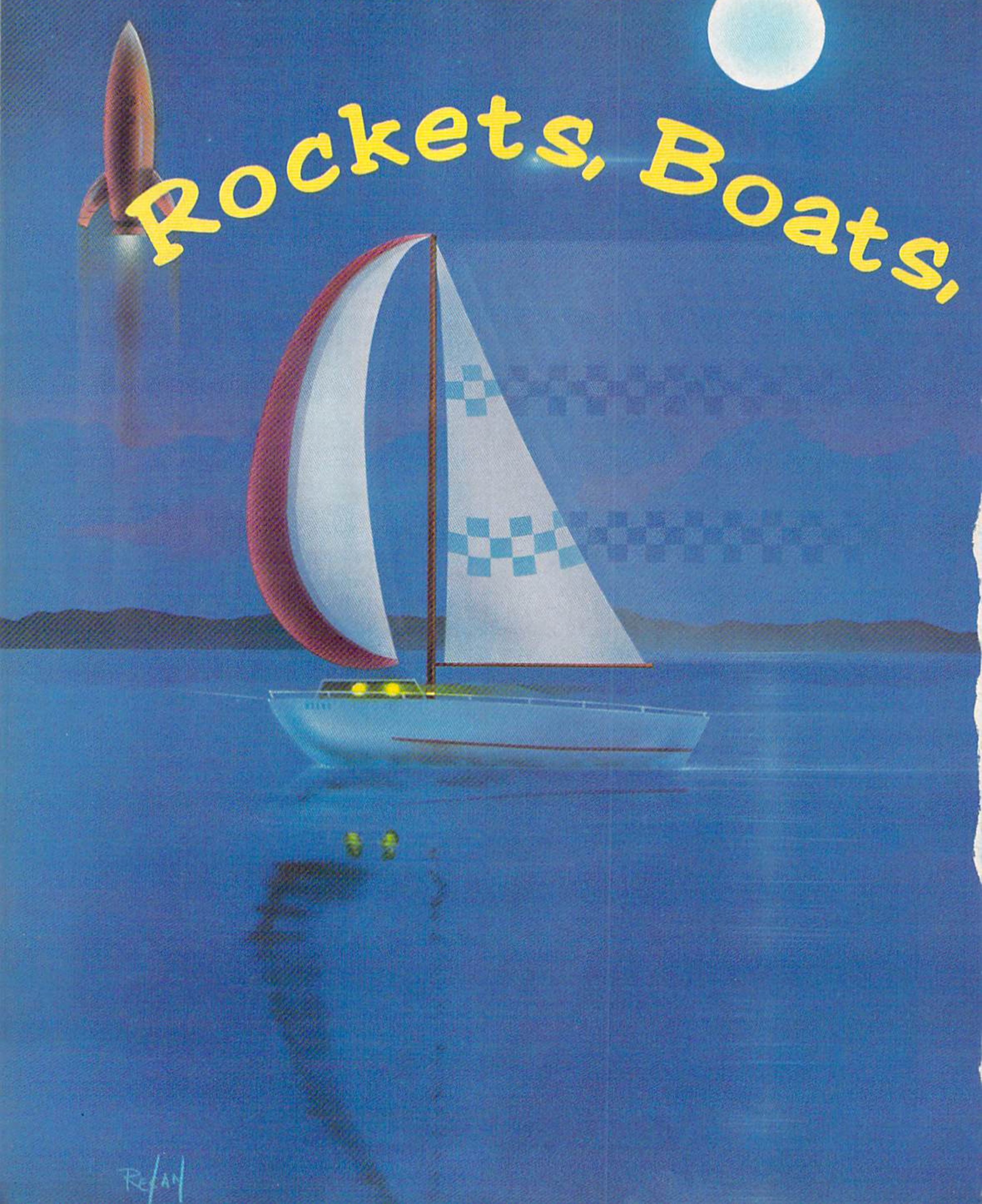
The program begins by initializing the disk and then requesting the name of the file to check. It will then give the start address in both hex and decimal and continue on to read in the file. It then gives the length of the file in bytes and finally the end address in hex and decimal.

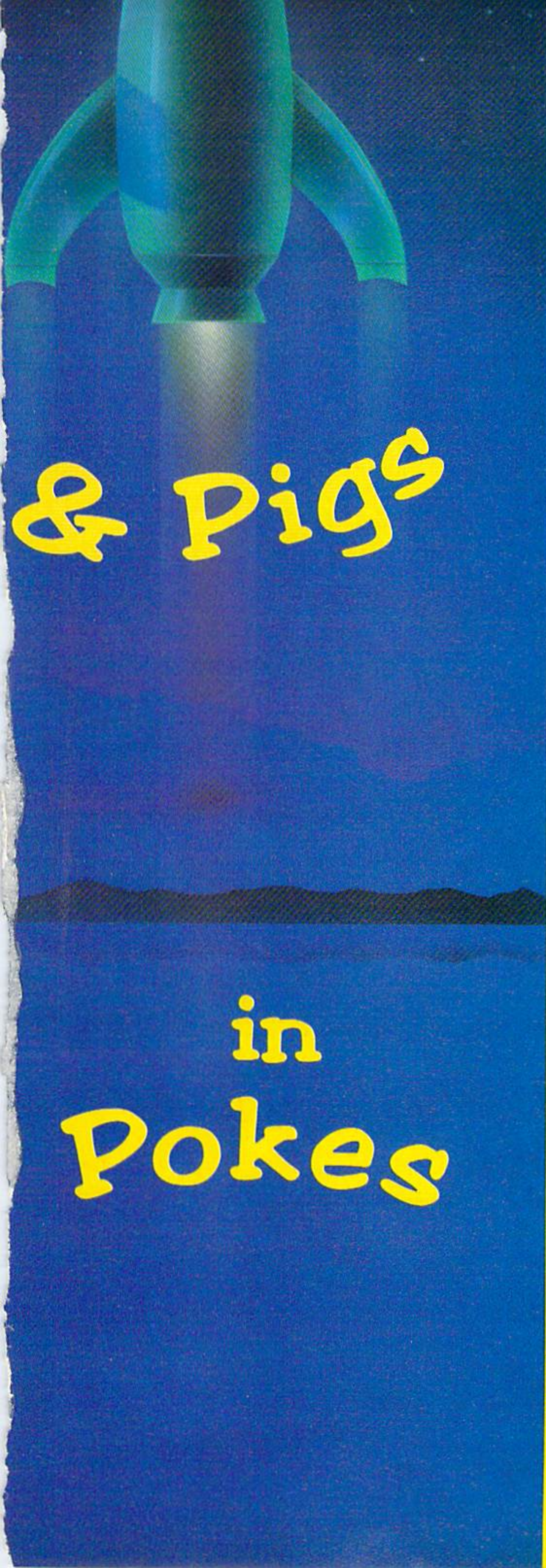
If you have another program residing in the cassette buffer, the machine language portion of this one can be easily relocated. ☐

SEE PROGRAM LISTING ON PAGE 109

How to use sprites effectively in your Commodore 64 action games — including *Pig in a Poke*, a simple children's game that uses sprite movement, shapechanging, and collision detection.

Rockets, Boats,





CREATING YOUR OWN GAMES ON THE VIC AND 64

In case you're one of those people who always wanted to grow up to be a swineherd, this is your lucky day. The program *Pig in a Poke,* included with this issue of *Ahoy!*, is just what you've been waiting for.

If, on the other hand, you're a nearly normal adult, you'll find that *Pig in a Poke* is not exactly challenging–it's about at the ability level of a four-year-old.

But that's all right. Four-year-olds need games, too.

And for those of you who are a bit older, the program is a clear demonstration of how to use sprites in game programming. It's very simple, using only one sprite and two sprite shapes, with the player able to control only horizontal movement. But by the time you're through studying the program, you'll know how to create sprites, move them, do shape-flipping, detect collisions, and make the sprite interact with the playfield and the player's instructions. And that's not bad for a simple program about a cute little pig trying to avoid the butcher's shop.

HOW TO PLAY

Once the program has finished setting up, you'll have a little pig on a grassy field. However, the robot swineherding machinery is in action, with moving fences sweeping across the field, trying to herd the pig toward the top of the screen.

You are the pig. All you have to do is run from side to side to avoid those fences. If they touch you, they'll push you upward, toward the top of the screen; but if you stay in the gaps between the fences, you'll be safe. Eventually, after fifty fences have gone by, the swineherds will give up and you'll be safe.

You control the pig by pressing the SHIFT key to move right and the COMMODORE key to move left. If you want to quit the game early, press SHIFT, COMMODORE, and CTRL all at the same time.

The game will end when fifty fences pass by; when you press SHIFT/COMMODORE/CTRL; or when the pig reaches the top of the screen.

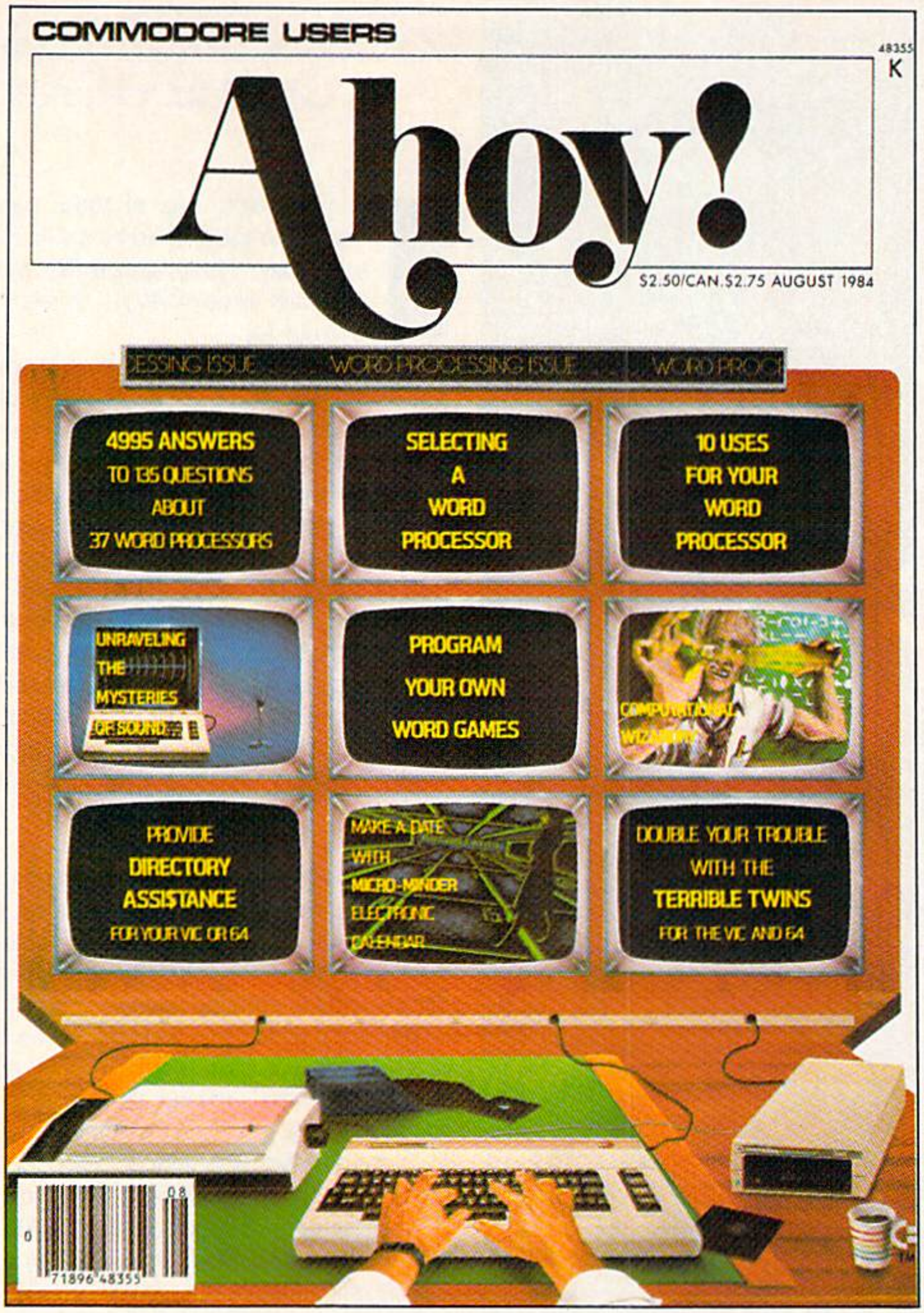
WHY ARE SPRITES SO GOOD?

The pig is, of course, a sprite. But there wasn't anything I did with that pig that couldn't be done with custom characters, was there?

Shapes and Sizes. Certainly the *shape* of the pig could have been duplicated with custom characters–six characters, in fact, to make the whole shape. That's because the pig is made out of 10 rows of dots, 23 columns wide. Since each character is eight dots by eight dots, it would take six whole characters to make the pig shape, two rows of three characters.

James Regan

By Orson Scott Card



HAS EVERYTHING!

SUBSCRIBE TO AHOY!

☐ Twelve Issues for $19.95 ($26.95 Canada and elsewhere)
☐ Twenty-four Issues for $37.95 ($49.95 Canada and elsewhere)

Name ____________________

Address ____________________

City __________ State __________ Zip __________

Send coupon or facsimile to:
ION INTERNATIONAL INC.
45 West 34th Street, Room 407, New York, NY 10001

The sprite can even be bigger than the pig. While the maximum width of a sprite is 24 dots, it can be up to 21 rows in height—more than twice as tall as my little pig.

Furthermore, the 64 allows you to change the size even more. You can double the sprite's width, or its height, or both. When you choose to expand the sprite, the VIC-2 video chip takes the shape from memory and doubles each dot; the pattern remains the same, but it is spread out over twice as many dots on the screen. If you double the width, the sprite then becomes 48 dots wide (though the *pattern* is still confined to 24 dots, each of which is doubled when it is displayed). If you double the height, the sprite becomes 42 dots high.

One-byte Registers. Sprite expansion is controlled with the two sprite expansion registers. Each bit of the single byte of the horizontal expansion register (at 53277) controls a single sprite. Sprite 0 is controlled by bit 0, sprite 1 by bit 1, and so on to sprite 7, controlled by bit 7. If a sprite's control bit contains a 1, its width will be doubled; if it contains a 0, the width remains normal.

For instance, let's say you want to double the width of sprites 0, 1, 2, and 7, but leave the other sprites alone. To turn on sprite 0, you need a 1 for bit 0, which has a decimal value of 1. Bit 1, when "on," has a decimal value of 2. Bit 2, when "on," has a decimal value of 4. You notice the pattern, of course—each bit's "on" value is double the "on" value of the bit before. Bits 3, 4, 5, and 6 have decimal values of 8, 16, 32, and 64—but since we aren't using them, we *won't* include those decimal values in our number. We *will* include the value of bit 7, however, which is 128.

Combine those, now: 1 + 2 + 4 + 128 = 135. To double the width of sprites 0, 1, 2, and 7, you only have to POKE 53277,135.

Several of the sprite control registers control all eight sprites with a single byte, each bit serving as an on-off switch for one of the sprites. In every case, bit 0 controls sprite 0, bit 1 controls sprite 1, and so on.

That's why in *Pig in a Poke*—as with any other program where I use sprites—I set up two arrays to control the on-off switches for the different sprites. The bit-set array, BS(0-7), contains the values 1, 2, 4, 8, 16, 32, 64, and 128. Then, when I want to turn on, or "set," the bit that controls, say, sprite 4, I only have to:

```
POKE 53275,PEEK(53275) OR BS(4)
```

The other array, BC(0-7), is the bit-clear array. It contains the values 254, 253, 251, 247, 239, 223, 191, and 127. Each value is a bit-mask: every single bit is "on" *except* the bit that controls the sprite in question. (The maximum value of a byte—all bits "on"—is 255. So the bit-mask for bit 0 is 255-1, or 254; the bit-mask for bit 3 is 255-8, or 247; the bit-mask for bit 7 is 255-128, or 127.)

Then, when I want to turn off, or "clear," the bit that controls sprite 4, I only have to:

```
POKE 53271,PEEK(53271) AND BC(4)
```

Changing Shapes. The sprite is also versatile in its shape. You can store, theoretically, 64 separate sprite-shapes in the 16K block of video memory. (Actually, you need to leave at least 1K for screen memory in text mode, plus room for as much character memory as your program needs.) Each of the eight sprites on the screen can take its shape from any one of the shapes you've stored in memory.

You could have all eight sprites look exactly the same by telling the VIC-2 to take their shape from the same pattern.

Or you could have one sprite cycle through dozens of different shapes in an animated sequence, with a single POKE for each change.

The sprite shape location registers aren't always in the same place, and their code is a bit complex. To understand how to use them, let's review some things we've covered in earlier columns.

Review of Video Blocks. As you remember, the VIC-2 video chip can only see 16K of RAM at a time. So all the graphics features have to be located in that 16K block: screen memory, character memory, sprite shapes, and bit-map memory. (The only exceptions are color memory, which is always at 55296-56319, and ROM charac-

ter memory, which is really located in a switchable bank at 53248-57343, but is seen by the VIC-2 at 36864-40959 when it is using the third video block, and at 4096-8191 when the VIC-2 is using the first video block. This is automatic, so once you select one of those blocks, the character set will be present for your use.)

The four possible video blocks are at 0K (from 0 to 16383), at 16K (from 16384 to 32767), at 32K (from 32768 to 49151), and at 48K (from 49152 to 65535). When the 64 powers up, it is using the first video block, starting at 0K. It puts screen memory at 1024 and sees the character set as if it were at 4096. This won't be too helpful when you're using sprites, however, because your BASIC program starts at 2048 and runs right through this whole block—there are only a few nooks and crannies into which you can tuck a sprite shape or two without bashing the BASIC program.

Often you'll use the second video block, at 16K, especially if your BASIC program is short enough to leave you some room. However, in that block you have to define your own character set. The third video block, at 32K, allows the VIC-2 to see character memory at 36K. You also can't use the last 8K of this block, from 40K to 48K, because that is BASIC ROM! But you still have 4K to contain screen memory and sprite shapes—and most of the time that will be enough, with plenty of room below 32K for a long BASIC program. The fourth block, at 48K, contains so much ROM that it's virtually unusable for video purposes.

The first two bits of the register at 56576 tell the VIC-2 which video block to use. First you AND the register with 252 to blank out those two bits, then OR it with the code for the video block you want. The code for 0K is 3; for 16K, 2; for 32K, 1; and for 48K, 0. (You need to diddle with the data direction register at 56578.) Here's how to select the 32K block:

```
POKE 56578,PEEK(56578)OR 3:POKE 56576,(P
EEK(56576)AND 252)OR 1
```

You also need to tell the VIC-2 where screen and character memory are, which is done at 53272; and tell BASIC where to find screen memory by POKEing the page number into 648. In *Pig in a Poke,* screen memory is right at 32K, which to the VIC-2 is 0K within the video block; BASIC uses the absolute page number, which is 128. Character memory is at 36K, which to the VIC-2 is 4K within the block. So the number POKEd into 648 is 128, and the number POKEd into 53272 is *character memory 1K boundary* plus sixteen times *screen memory 1K boundary,* or 4+16*0. (For those of you who grew up with the New Math, that adds up to 4. A lot of work just to get a dumb little one-digit number, isn't it?)

Now that we've told the VIC-2 where to find all this stuff, where are those sprite location registers? The sprite location registers are *the last eight bytes of the 1K block where screen memory resides.* That's why we had to go through all this rigmarole. You have to know where screen memory is to find out where the sprite location registers are. The address is *video block address* plus *screen memory 1K boundary times 1024* plus *1016.* In *Pig in a Poke,* that's 32768+0+1023, or 33791.

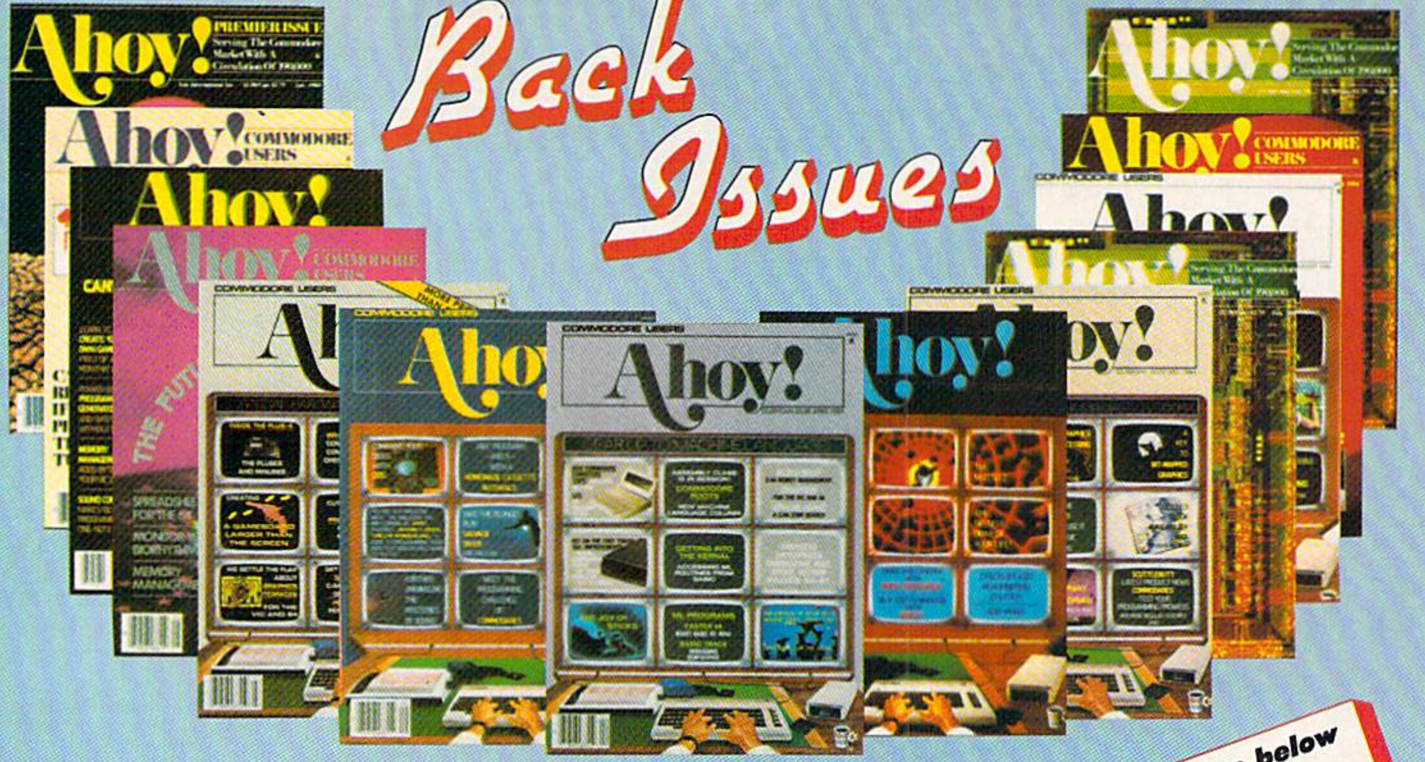
Each sprite shape location register contains a number from 0 to 63, representing one of the 64 possible 64-byte blocks of memory within the video block. For instance, a 10 tells the VIC-2 to look for the block starting at address *video block* plus *ten times 64.* A 55 would tell the VIC-2 to look at *video block* plus *55 times 64.* Since our video block starts at 32768, sprite shape block number 10 (the eleventh block, since the first block is numbered 0) starts at 32768+10*64, or 33408.

However, since screen memory in our program starts at 32768, and uses 1000 bytes of memory, that sprite block is *within* screen memory. Now, this doesn't cause the VIC-2 any trouble. It will merrily display *both* the screen *and* the sprite, using overlapping regions of memory. VIC-2 doesn't care. But *you* probably will. As the screen display changes, so will your sprite shape!

A Small Example. Let's throw a sprite shape onto the screen showing just how that works. Power up your 64, and enter the following program lines in direct mode. Notice that we will *not* change the video block—we're using the screen as you normally have it.

```
POKE 53269,1
```

Back Issues



Use the form below to order while limited supplies last!

ISSUE #1—JAN. '84 $4.00
64 v. the Peanut! Telecommunications! Tales of the Commodore! And ready to enter: Multi Draw 64 Graphics System! Interrupt Music Maker/Editor! A Peek at Memory! Sequential Files!

ISSUE #2—FEB. '84 $4.00
Illustrated tour of the 1541! Artificial intelligence! Synapse's Ihor Wolosenko interviewed! And ready to enter: Music Maker Part II! Screen Manipulation! Night Attack! Relative Files!

ISSUE #3—MAR. '84 $4.00
Anatomy of the 64! Printer interfacing for VIC & 64! Educational software series begins! And ready to enter: Address Book! Space Lanes! Random Files on the 64! Dynamic Power!

ISSUE #4—APR. '84 $4.00
Petspeed and Easy Script tutorials! Printer interfacing continues! Laserdisc! And ready to enter: Apple Pie! Lunar Lander! Name that Star! Lower case descenders on the 1525 printer!

ISSUE #5—MAY '84 $4.00
Future of Commodore! Inside BASIC storage! Memory management on the VIC & 64! Guide to spreadsheets! And ready to enter: Math Master! Air Assault! Biorhythms! VIC Calculator!

ISSUE #6—JUNE '84 $4.00
Game programming column begins! Program generators! Rupert on inputting! Memory management continues! And ready to enter: Post Time for the 64 & VIC! Alpiner! Sound Concept!

Sorry...Issues #1, #2, and #7 sold out! Reprint editions (programs and feature articles only) available for $4.00 each.

ISSUE #7—JULY '84 $4.00
MSD dual disk drive! Database buyer's guide! Training your cursor! Screen displays! File Sleuth! Users Groups! And ready to enter: Renumbering! Checklist! Math Defender! Brisk!

ISSUE #8—AUG. '84 $4.00
Choosing a word processor! Computational wizardry! Creating your own word games! Sound on the 64! And ready to enter: Micro-Minder! Directory Assistance! The Terrible Twins!

ISSUE #9—SEPT. '84 $4.00
Program your own text adventure! Build a C-64 cassette interface! Video RAM! And ready to enter: Salvage Diver! DOS! Sound Explorer! The Castle of Darkness! Base Conversions!

ISSUE #10—OCT. '84 $4.00
C-64 graphics programs! Bit-mapped graphics! Joystick programming! And ready to enter: VIC 40 Column Operating System! BAM Read & Print! Emerald Elephant! Lawn Job!

ISSUE #11—NOV. '84 $4.00
Music programs & keyboards for the 64! Graphics feature continues! And ready to enter: PTE word processor! Block Editor! Alternate Character Set for the 64! The Tunnel of Tomachon!

ISSUE #12—DEC. '84 $4.00
Buyer's guide to printers! 1525 printer tutorial! Custom characters! User Guide to KMMM Pascal! Diving into BASIC! And ready to enter: Construction Co.! Space Patrol! Cross Ref!

ISSUE #13—JAN. '85 $4.00
VIC/64 OS exposed! Sprites! 1541 device # disconnect switch! Ghostbusters! And ready to enter: Ultra Mail! Music Tutor! Alice in Adventureland! Midprint! To the Top! Tape/Disk Transfer!

ISSUE #14—FEB. '85 $4.00
Printer interfacing! Multicolor sprites! Modems! Bulletin boards! Theory of game design! And ready to enter: Futurewar! Fontasia! VIC Eraser! Insurance Agent! Flankspeed! Telelink 64!

ISSUE #15—MAR. '85 $4.00
Making multiscreen gameboards! Inside the Plus/4! Commodore DOS! And ready to enter: Old Routine! Programmable Functions! Automatic Line Nos.! Home Budget! Salmon Run!

ISSUE #16—APR. '85 $4.00
Assembly language column begins! Programming the joystick! 1541 disk drive alternatives! And ready to enter: Hop Around! Faster 64! Booter! Elecheck! BASIC Trace! Space Hunt!

ISSUE #17—MAY '85 $4.00
Disk drive enhancements! Install a reset switch! Assembler escapades! And ready to enter: Super Duper! Two-Column Directory! DSKDU! Raid! DOS Plus! Font Editor! Tile Time!

ISSUE #18—JUNE '85 $4.00
Music & graphics entry systems! How modems work! Inside the 6510! And ready to enter: Quad-print! Mapping 4.4! Towers of Hanoi! Speedy! Duck Shoot! The 6510 Simulator!

ISSUE #19—JULY '85 $4.00
PROM programming! 3-part harmonies on VIC/64! Speeding pixels! And ready to enter: Auto-Append! Script Analysis! Wizard of Im! Lucky Lottery! Brainframe! Etch! Printat!

ISSUE #20—AUG. '85 $4.00
Inside the 128! Read-world simulations! Sound effects! And ready to enter: Windows! Formatter! Sound-a-Rama! Screen Dump! Selectachrome! Disintegrator! Fidgits! Gators N Snakes!

Use coupon or facsimile. If ordering more than three issues, list choices on separate sheet.

If you have a modem and want more complete information on any of the back issues listed, call *Ahoy!*'s Bulletin Board Service at 718-383-8909.

Ahoy!
BACK ISSUES
ION INTERNATIONAL INC.
45 West 34th Street
Suite 407
New York, NY 10001

Please Send Me The Following:
____ Copies of issue number ____
____ Copies of issue number ____
____ Copies of issue number ____

Enclosed Please Find My Check or Money Order for $____
(Outside the USA please add $1.00 for every copy)

NAME ____
ADDRESS ____
CITY ____ STATE ____ ZIP CODE ____

This enables sprite 0.

```
POKE 53248,175:POKE 53249,135
```

This puts the sprite at horizontal coordinate 175 and vertical coordinate 135. That's just about in the middle of the screen.

You'll notice that at this point the sprite is a rectangular display of garbage. But it isn't just *any* garbage – it is the garbage pointed to by the sprite shape location register for sprite 0. Remember, this is found at byte 1016 from the start of screen memory. The default screen memory for the 64 is at 1024, so that the sprite 0 location register is at 1024+1016, or 2040.

Upon powering up, my machine puts a 32 at 2040. This means that until I change it, sprite 0 will display whatever is in memory at 0+32*64 (video block 0K plus 64 times sprite shape location code 32), or 2048. It just so happens that this is the address of the beginning of the BASIC program area. Just enter a few meaningless lines, like 100 REM and 200 REM THIS IS JUST A TEST. With each line you enter, you'll see the sprite shape change as BASIC builds a program in the sprite shape block. (If your 64 shows something else, POKE 2040,32 and enter or delete program lines.)

Now let's move the sprite shape block somewhere else. In direct mode:

```
POKE 2040,2
```

Right in the middle of the sprite you'll see some flickering dots. These are the constantly changing bytes at 167 and 168, where the operating system maintains a clock.

Can't see it clearly enough? Try using the horizontal and vertical expansion features:

```
POKE 53271,1
```

That makes it nice and tall.

```
POKE 53277,1
```

And now it's fat, too. If you POKE zeroes into the same location, it becomes short and skinny again.

Bits to Make Shapes. How does the VIC-2 read the sprite shape matrix? Basically, it reads the matrix the way we read books, starting at the upper left-hand corner of the sprite shape and reading across to the right; when the line ends, it drops down to the left-hand edge of the next line. Each of the 24 dots on each row is represented by one bit in the sprite shape block. The first byte of the block contains the bits that control dots 0-7 of the first row of the shape; the second byte, dots 8-15; the third byte, dots 16-23. That's all for the first row, so the next byte in the shape block controls dots 0-7 of the second row of the shape, and so on, until the 63rd, or second to last, byte of the block controls dots 16-23 of the row 20 (the 21st row, since we began with row 0). The very last byte of the 64-byte sprite shape block controls nothing at all – the VIC-2 ignores it, and so can you.

The example program *Basic Sprites* shows how to create sprite shapes quite easily. The routine in lines 800-810 reads some DATA statements that contain the sprite shapes as *strings*. This is so you can design the shape right there on the screen without any complicated graph paper calculations. Each string represents one row of the sprite shape, with asterisks representing "on" bits and periods representing "off" bits. The routine breaks the string into three 8-character chunks, each of which corresponds to a byte of the sprite shape block. Byte by byte, row by row, it reads, converts, and POKEs in the sprite shape.

Since we're only using two sprites and three sprite shapes, they are being POKEd into the cassette I/O buffer at 832 (sprite shape block 13), 896 (block 14), and 960 (block 15). Thus they don't interfere with BASIC's program storage above 2048. We aren't doing any fancy changing of the video block for this program.

And once it's SAVEd, you can go ahead and change the sprite shapes in the DATA statements starting at 810. Just remember that every string has to contain 24 characters, one for each bit of the sprite's width, and there must be 21 strings per sprite, one for each row of the sprite shape block.

Movement. With character graphics we were able to move objects around the screen with some good speed,

using the PRINT command. However, since a character can only be in one 8-by-8 grid or another, each movement had to be at least eight dots at a time. It looks jumpy and is hard to watch.

Sprites can move much more smoothly, since they aren't mapped onto the text screen. They can move one dot at a time, vertically or horizontally, as the program *Basic Sprites* showed. (The program *Pig in a Poke* moves the pig two dots at a time horizontally, to increase the speed; that's about as big a jump as you can take without losing smoothness. It also moves eight dots at a time vertically, so it will always stay in alignment with the character-graphics fences.)

Each sprite has its own horizontal position register and vertical position register. These registers start at 53248. They proceed in order, starting with sprite 0's horizontal position register at 53248, sprite 0's vertical position register at 53249, sprite 1's horizontal register at 53250, sprite 1's vertical register at 53251, and so on.

The registers contain coordinates that correspond with dots on the screen. They always specify the location of the dot in the *upper left-hand corner* of the sprite's shape matrix. This has nothing to do with whether that dot is turned on or not—even if your whole sprite shape is crowded down at the bottom of the shape block, the VIC-2 will still use the position registers to calculate the location of the upper left-hand corner of the sprite shape block.

A sprite can be half on and half off the display area. Whatever part of the sprite is past the edge of the display area and into the screen border will seem to move "behind" the border—that is, the dots that are off the edge of the display area are simply not displayed.

That's why a sprite with a horizontal or vertical position of 0 is completely off the screen—it's "behind" the border. The leftmost position a sprite can have and still be completely visible on the screen is 24. The highest position is 50. The first position to the right that will have the sprite completely *off* the screen is 344, and the first position on the bottom that will make the sprite completely disappear is 250. (When the sprite *starts* to disappear on the right-hand edge or the bottom of the screen depends on whether or not it has been expanded in either direction.)

You'll notice, though, that the horizontal position can go well over 255, which is the highest number that can be POKEd into a single byte of memory. This means that once the sprite reaches horizontal position 255, the horizontal position register can't possibly move it any farther to the right—the number 256 cannot be contained in the register.

The VIC-2 gets around this by putting a sort of International Date Line at position 256. When the sprite reaches that horizontal position, a flag bit is set Somewhere Else, and the horizontal position register starts over at 0. So to move across the International Date Line, you POKE the horizontal register with 253, 254, 255, then set the flag bit, and go on with 0, 1, 2, and so on.

The flag bits for all eight sprites are in a single register at 53264. Sprite 0's flag bit is bit 0, and so on, as with all the other one-byte sprite control registers. If a sprite's flag bit is clear (contains a 0), then the VIC-2 will count the sprite's horizontal position from far left of the screen; if a sprite's flag bit is set (contains a 1), then the VIC-2 will count the sprite's horizontal position from the International Date Line right on the screen.

To handle the horizontal and vertical ranges smoothly, both the *Basic Sprites* example program and the *Pig in a Poke* game use three arrays. The array VV*(n)* contains all the "valid" vertical locations. In *Pig in a Poke*, there are only 20 valid vertical positions, at every eighth row starting with row 53. So in lines 900 and 906, this array is set up to have 20 elements, from 0 to 19, which contain, in order, the vertical position codes 53, 61, 69, 77, 85, 93, 101, 109, 117, and so on to 205. Now the program only has to keep track of which of the twenty valid vertical positions the sprite should be on, using the variable VP. Then it sets the vertical positions with POKE 53248,VV(VP).

A similar thing is done with the horizontal position, except that two arrays are used. The first array, HH*(n)*, contains all the valid horizontal position codes for the horizontal position register. The second array, HB*(n)*, contains the flag bits for the register at 53264. Each time the sprite is moved horizontally, *both* numbers are POKEd

into the control registers.

Why both numbers *every* time? For smoothness. You could use an IF-THEN test to change the flag bit only when the sprite is actually crossing the International Date Line. But in a BASIC program, with its slow POKE statement, that will mean that the sprite hesitates for a moment. By POKEing those values every time, overall movement will be slower, but there's no hesitation. (However, in either case there will be a bit of blinking at that point, because for the moment between POKEing the horizontal register the POKEing the flag-bit register, the sprite will be completely offscreen. This is avoidable only in machine language.)

Another solution, of course, is to keep your sprites from crossing the International Date Line at all. Display something else to the right of the line, and keep the sprites always to the left. No flicker then!

Three Dimensions. The priority register at 53275 controls whether the sprite moves *in front* of objects on the screen or *behind* them. Each bit of the byte at 53275 controls the corresponding sprite. If sprite 0's bit, bit 0, is *set* (equals 1) then the sprite will seem to move *behind* objects on the screen. That is, the sprite will be visible whenever it is in front of the background color of the screen, but every dot that is on the same spot as a dot in a character shape on the screen will be invisible—the character shape will seem to be in front of the sprite. If bit 0 is *clear* (equals 0) then the sprite will move in front of the character shape as well as the background.

This lets you create the illusion of three dimensions on the screen. However, setting and clearing individual bits of the priority register is so cumbersome that in a BASIC program you will probably want to set the priority once and leave it alone. Only in machine language can you fiddle it on and off without slowing down program execution hopelessly.

Collisions. For game programming, one of the most important features of sprites is that the VIC-2 automatically keeps track of whether sprites have bumped into each other or into any character shapes on the screen. Two collision registers keep track of whether each sprite has been involved in a collision since the register was last checked. To see if sprite 4 has bumped into another sprite, examine bit 4 of the sprite/sprite collision register at 53278. To see if sprite 7 has bumped into any of the character shapes (or bit-map designs) on the screen, examine bit 7 of the sprite/foreground collision register at 53279.

When does a collision take place? Only when an "on" dot in the sprite shape occupies the same screen position as an "on" dot in another sprite or an "on" dot in a character shape. If the transparent portion of a sprite's shape matrix passes over another sprite or a character, nothing happens to the collision registers. Only when the dots that are actually displayed come into "contact" is a collision reported.

The collision registers "remember" every collision until your program PEEKs (or LDAs or CMPs, in machine language) the collision register. This means that you should always clear the collision registers by PEEKing them once at the beginning of the program, before you start checking them in earnest. Also, it means that you can't PEEK the sprite/sprite register to check for a collision for sprite 0 and then check it again for a collision for sprite 1—when you checked it the first time, it was erased! That's why you'll want to put the byte in a variable, like this: Z=PEEK(53278). Then you can check the variable Z to test for as many collisions as you want.

Unfortunately, these registers don't tell you *which* sprite or character shape you bumped into. Of course, if only sprites 0 and 1 in the sprite/sprite collision register report a collision, you can bet they bumped into each other. But if sprites 0, 1, and 2 report a collision, you don't know if both 0 and 1 bumped into 2, but not into each other, or if they all bumped into each other—and it can get even more complex.

To know *which* sprite was bumped, you either have to track the location of every single sprite (in which case a BASIC program will crawl along at the speed of a student driver), or you have to decide that *all* sprite collisions have the same effect, so you don't *care* which sprite bumped which.

That's what *Pig in a Poke* does. Since there's only one sprite onscreen, there can't be any sprite/sprite collisions at all. And *any* collision with character patterns has ex-

actly the same effect—bouncing the pig toward the top of the screen.

If I had wanted to have characters on the screen that the pig could touch at will, then I would have had to use *sprites* for the fences moving up the screen. In that case, sprite/foreground collisions would have been ignored, and only sprite/sprite collisions would have mattered. Furthermore, it wouldn't have mattered *which* sprite the pig bumped into, because *any* sprite would have the same effect.

Any time it doesn't matter *which* sprite the player's sprite bumped into, collision handling is simple. When it does matter, it becomes complex and generally has to be handled in machine language.

Other Features. Sprites have several other possible features, which I'll discuss in later columns, like multicolor display. These features have their own registers, which are mentioned in the Sprite Memory Map but not explained here.

The sprites' colors are controlled by eight sprite color registers; however, the 64 automatically assigns eight different colors to the sprites when the machine is turned on, and the example programs don't change them. If you want to, however, just POKE into them the number of the color you want, from 0 to 15, just as you do with the background color register at 53281. The sprite color registers are located at 53287 through 53294.

Also, there are advanced techniques like animation (shape-flipping), rotation, and complex sprites, which we'll go through in the next couple of months, including an interrupt-driven machine-language routine that will allow you to speed up sprite movement drastically while still controlling it all from BASIC.

SPRITE MEMORY MAP

For easy reference, here are the sprite-control registers in the Commodore 64. For explanations, see the text of the article.

Floating Memory Locations

1016-1023 after the start of screen memory (default 2040-2047): The eight **shape location registers**. (If you're using bit-map mode, they can be found at bytes 1016-1023 after the beginning of bit-map color memory.)

0-63: The codes for the possible location of each sprite shape block within the video block. The actual address is found by multiplying this code by 64 and adding the result to the absolute address of the start of the video block. The first 63 bytes of the sprite shape block determine which dots will be "on." Every 1 bit causes a dot to be displayed; a 0 bit allows the playfield display to show through.

Permanent Control Registers

53248, 53250, 53252, 53254, 53256, 53258, 53260, 53262: Horizontal position registers for sprites 0 through 7, in order.

53249, 53251, 53253, 53255, 53257, 53259, 53261, 53263: Vertical position registers for sprites 0 through 7, in order.

53264: Flag-bit register for horizontal position of all eight sprites.

53269: Sprite enable register. (255 enables all sprites; 0 disables all sprites.)

53271: Sprite vertical expansion register. (255 expands all sprites.)

53272: Screen and character memory location register. Where screen memory is located determines where the sprite shape location registers are located. The high four bits are significant for screen memory location:

> *screen location*=(PEEK(53272)AND 240)/16+*video block starting address*

53275: Sprite priority register. (0 means all sprites appear in front of foreground; 255 means all sprites appear behind foreground.)

53276: Sprite multicolor enable register. (255 makes all sprites display using multicolor option.)

53277: Sprite horizontal expansion register. (255 expands all sprites.)

53278: Sprite/sprite collision register. (0 means no collisions; PEEKing this location clears the register.)

53279: Sprite/foreground collision register. (0 means no collisions; PEEKing this location clears the register.)

53285: Sprite multicolor register 0, controlling the color of 01 bit-pairs.

53286: Sprite multicolor register 1, controlling the color of 11 bit-pairs.

53287, 53288, 53289, 53290, 53291, 53292, 53293, 53294: Sprite color registers for sprites 0-7, in order. Default colors are white, red, cyan, purple, green, blue, yellow, and medium gray. (In multicolor mode, these registers control the color of 10 bit-pairs.)

53276: Video block selection register. Bits 0 and 1 determine which video block the VIC-2 chip will read.

HOW PIG IN A POKE WORKS

In th meantime, though, let's go through *Pig in a Poke* section by section in order to see exactly what's going on.

The Variables. One of the most confusing things in studying somebody else's program is to try to remember what all the variables mean. So here is a list of the major variables used by both *Pig in a Poke* and *Basic Sprites*:

VB	Address of the video block
SB	Address of screen memory
BB	The screen memory page number (used by BASIC
LT(0-7)	The address of each sprite's shape location register, located starting at byte 1016 after the start of screen memory.
CT(0-7)	The address of each sprite's color register.
HT(0-7)	The address of each sprite's horizontal position register.
VT(0-7)	The address of each sprite's vertical position register.
HR	The address of the flag-bit register for horizontal movement across the "International Date Line."
ES	The address of the sprite enable register.
VE	The address of the vertical expansion register.
HE	The address of the horizontal expansion register.
PR	The address of the priority register.
EM	The address of the multicolor enable register.
MR	The address of the first multicolor color-selection register.
CS	The address of the sprite/sprite collision register.
CF	The address of the sprite/foreground collision register.
BS(0-7)	The bit-set values for each sprite.
BC(0-7)	The bit-clear values for each sprite.
ST*(n)*	The codes for the location of each sprite shape block; these numbers are POKEd into the sprite location registers at LT*(n)*.
NS	The number of sprite shapes, minus 1.
VV*(n)*	The vertical position array; these values are POKEd into the vertical position registers at VT*(n)*..
HH*(n)*	The horizontal position array; these values are POKEd into the horizontal position register at HT*(n)*.
HB*(n)*	The horizontal flag-bit array; these values are POKEd

Continued on page 75

FISH MATH

for the

VIC 20

By Kevin Dewey

In this educational game for children ages eight through twelve, the player is in control of a group of ten fishes that must be led to the top of the stream safely and with the right amount of food in their tummies. The player figures out how much food each fish requires by working out a simple multiplication problem at the bottom of the screen. He then leads the fish up the stream, gobbling up food as he goes, and avoiding the shark known as Gums. Contact with Gums makes you into shark-bait.

CONTROLLING THE FISHIES

Controlling the fishies is a very complex process to learn, but very easy once you get the hang of it. There are only two keys—for right and left. The CRSR keys are used, CRSR UP/DOWN moving the fishy left and CRSR right/left moving the fishy right. Doesn't sound too complex, does it? But your fishy is always moving up, almost automatically, except when you are pushing down a CRSR key. So if you have a lot of food to eat, you'd better not go too long without moving right or left, or before you know it, you'll be on the top of the screen, with far less food than the fish requires. This problem can be solved, however, by scrolling off the side of the screen and moving down. You only move down on the right side. If you scroll off the left side, you move up. So if you're too close to the top of the stream with not enough food, you can head back down if you keep going right.

Another key that comes in handy is SHIFT/LOCK. This pauses the game, allowing the player to think over the multiplication problem presented, or the best way to avoid Gums, whichever is most pressing at the moment. Play is resumed by depressing the key.

SCORING

For every fish brought to the top safe and well-nourished, a point is added to your score. For every fish either overfed or underfed, a point is taken away. If Gums gets a fish, the game ends.

A FEW WORDS ON EDUCATION

The educational emphasis in this game is on teaching basic multiplication facts. I had originally chosen to make it an addition game, but chose not to when I realized the arcade element of the game might become frustrating for the younger set of children.

Instead, I chose multiplication because older children can appreciate the challenges of an arcade game while learning, instead of becoming frustrated by it. The game really is quite hard to master, and though children may not realize it, they are being drilled in multiplication skills as they play. □*SEE PROGRAM LISTING ON PAGE 110*

AHOY! DOCK

A Mini Database Manager for the C-64

By Glenn Lumpkins

How many times has this scene been played? Your latest issue of *Ahoy!* arrives with the mail. You flip through the pages thinking: boy-these-are-great-programs, the ads, all-those-helpful-hints, and the many other inspiring articles. I'll read this issue and type in the programs when I have time to really concentrate. For now, I'll make some mental notes about this issue. You now add this issue to the month's stack of magazines which sit on top of last month's stack, which sit on top of the previous month's stack, and so on.

When was the last time you went to that stack of material needing a routine, a helpful hint, an item, or an article on a specific subject? Pulling issues out of the stack and scanning their indexes, you think to yourself: "One of these days I've just got to get this stack organized. But the issues keep piling up higher and there seems to be no end to it. There must be a magnetic quality to the computer desk, as everything seems to gather there. Maybe tomorrow I'll get started."

HELP IS ON THE WAY!

I wrote *Ahoy! Dock* because I receive and save several computer publications each month. More than once I have tried to catalog and file these issues, all to no avail.

Now with *Ahoy! Dock* and a 1541 disk drive, I have an index of every article in my issues that I might ever want to use or read. *Ahoy! Dock* allows me to add to my file at any time, search for a program or articles, delete old items, modify the file, print or list the files, and sort the file on multiple fields. It now takes minutes instead of hours to find the information I need and my desk is much neater, thereby allowing me more computer time.

Ahoy! Dock is simply a mini-database manager designed for use with an auxiliary storage device and a floppy disk. It will allow you to index information according to 14 categories, with each category containing 500 independent records. With its simple menu operations, the program is as easy to use as pressing the function keys.

When you run *Ahoy! Dock*, you'll see a main menu. Step-by-step instructions are asked for by each of the menus, with every attempt made to prevent an incorrect entry. If all else fails, at any point of entry a "/" will allow you to escape back to the main menu.

When you select an option from the menu, a submenu of categories will be read to the screen for you to choose from. When you select the ADD option and a category, the program will direct the computer to read the error channel from the disk drive to see if that category exists. If the category doesn't exist, you are asked to create the file or return. Choosing the create option will initialize that file.

With just a little of your time, you'll soon have a comprehensive disk file that will index all the magazine articles and programs you wish.

Ahoy! Dock was written for computer magazines. You can easily alter it for any type of publication by changing the data subject categories and the program lines. □

SEE PROGRAM LISTING ON PAGE 103

FLOTSAM

Because upon starting the Commodore, RND(1) always produces the same series of random numbers, and because RND(0) is rather suspect, I use one of the following lines in programs that use random numbers:

```
100 R = RND(1): GET A$: IF
 A$="" THEN 100
```

or

```
100 R = RND(1): IF (PEEK(5
6320)AND15)=15 THEN 100
```

Depending upon the length of the pause, line 100 will generate a varying number of random numbers, which changes the starting point for the series of random numbers used in the program. The more often the program can zero line 100, the better. —Jacqueline L. Callaway
Orange Beach, AL

I would like to address an article in the May issue of your fine magazine: *Disk Spinners, Part II* by Morton Kevelson. I have used all three of the referenced devices. I purchased the 1541 Flash after reading the article and cannot offer enough praise for this fine product.

I have the updated model with the switch on the disk drives (2). I have been using the Flash for about a month and have found only two games that will not work with the Flash installed.

I use word processors and *PFS (File & Report)*. I have found that with the *Flash* installed it is a pleasure not waiting between searches during file update and printing.

On the Commodore I have, the Kernal ROM was soldered in and did require the tricky removal of the old Kernal to install the socket furnished with the kit; after this was done the rest of the modification went without any problems.

I would like to thank Mr. Kevelson for his article and Skyles Electric Works for their product.
—David J. Farrell Sr.
Enfield, CT

I recently purchased the *Fast Load* cartridge from Epyx. One of the first programs that I tried *Fast Load* on was Microsoft's *Multiplan* for the C-64. *Multiplan* is a superb spreadsheet, but frustratingly slow. When I attempted to save some changes to my data disk, *Fast Load* restructured my files so that I could not retrieve them.

I called Epyx to find out what happened. The gentleman I talked to said the problem was relative files. His belated advice was "don't use *Fast Load* on any program containing relative files."

In all the reviews that I have read in your magazine and others, nowhere is this problem mentioned. And nowhere are there any warnings about potential problems like this.

Please address this problem for my benefit and the benefit of all your readers. —Edward S. Champa
Indianapolis, IN

Scuttlebutt would be even better if it incorporated, though in abridged form, the same thoughtful judgment that's found in your *Reviews* section. Any chance of that? *—Mel Getty*
Orono, ME

Not if we want to keep Scuttlebutt *the most current—as well as complete—Commodore news section on the stands. We announce new products as soon as we learn of their existence. Waiting for review samples would make our news section yesterday's papers.*

We incorrectly listed the address and phone number of Navarone Industries, Inc. on the back cover of our August issue. Correct address and phone number are:
Navarone Industries, Inc.
19968 El Ray Lane
Sonora, CA 95370
Phone: 209-533-8349

FASTNEW

A 20-Second Formatter for the 1541

By Don Lewis

Fastnew is a fast disk formatting program. The 1541 has a built-in command for formatting, i.e. N0:(diskname,id). So why a special program? Because *Fastnew* improves on the 1541 resident formatter in three important areas:

a. Speed. The normal format takes about 90 seconds. *Fastnew* formats a disk in 20 seconds.

b. Reliability. The normal format routine includes a verify. Unfortunately, this verify has a bug in it which allows some format errors to go undetected. *Fastnew* has a correct verify, insuring error-free formatting.

c. NO Head Bang. The normal format kills 1541's. Every time you use the normal format, the read/write head is BANGed against a hard stop in the drive. Eventually this will throw the disk out of alignment, and cost you an expensive repair. *Fastnew* never bangs the read/write head when formatting.

Fastnew consists of three sections, a BASIC portion and two machine language (ML) routines. The BASIC code handles screen output and user input. One ML routine is a fast memory mover used to transfer code from the 64 to the 1541. The other ML section is the formatter proper. It is transferred to the 1541, then executed using an M-E command. This code relies heavily on subroutines in the 1541 ROM's to perform the formatting.

When typing in *Fastnew*, be sure to double (and triple) check the data statements at lines 4000-4999 and 5000-5999. Make sure you save a copy to disk before running the program. Until you get a completely working *Fastnew*, be prepared to quickly turn your disk drive off. Typos in the data statements can cause the 1541 to really go wild.

Source code for the 1541 resident code is available for $8.00 postpaid from the author (see address below). Careful study of the source code will suggest all sorts of possibilities for custom formatting, such as varying data or header block lengths, extra tracks or sectors, and recording density shifts. Custom formatting has been where most copy protection schemes originate.

Fastnew is faster because it avoids a lot of the work the normal format does, without compromising integrity. When formatting a disk the 1541 normally does quite a bit of work. First it writes 10250 $55 bytes (non-synch marks) to the disk. Then it writes an alternating pattern of $55 and $FF (synch marks) bytes to the disk. It then reads the track and counts how many synch and non-synch marks were actually written on the track. From these counts, it calculates the optimal inter-sector gap length. The goal is to divide the track into evenly spaced sectors. The actual number of bytes written out as the inter-sector gap depends on the length of the track (track 35 is shorter than track 1), the number of sectors on the tracks, and the speed of the disk drive, which may vary from day to day. After calculating the inter-sector gap, the actual formatting is done. The header block and a dummy data block are written out for each sector on the disk. The Read/Write head is then stepped to the next track and the process repeated.

In practice all this work isn't necessary. There is no reason to try to get an even sector distribution around the track, provided there is enough space for all the sectors. It is important, however, to have some gap between the end of one sector and the start of another. If a gap did not exist, you'd run the risk of having a data block write into the header block of the next sector, causing a disk error number 20. This could happen if the speed of the disk drive varies from day to day.

Rather than go through all the calculations described above, *Fastnew* assumes that a gap of 8 bytes between sectors is sufficient. This is based on an examination of the inter-sector gaps on a number of normally formatted disks.

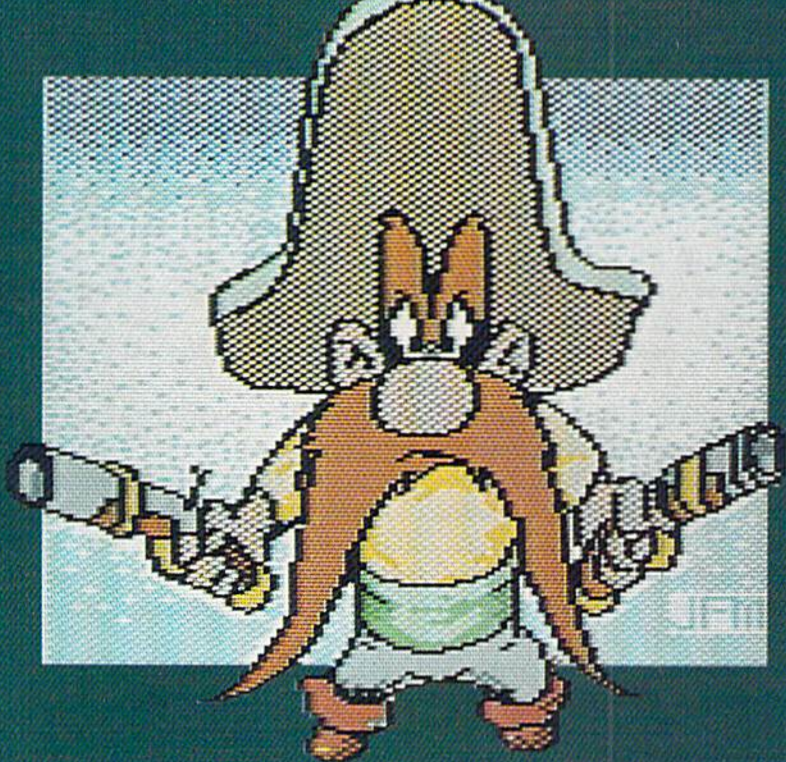
Head bang occurs when the R/W head is pulled back to track 1 by the normal format routine. The normal format assumes that the head is as far away as possible and steps the head back 45 tracks–BANG!! However, provided that the disk drive has accessed at least one properly formatted disk, the current position of the R/W head can always be found at memory location $22. *Fastnew* reads this location and calculates the necessary steps to pull the head to track 1 with NO bang.

As I said, *Fastnew* makes one assumption, that the disk drive has previously accessed a properly formatted disk. (A small price to pay for extending the life of your 1541 disk drive.)

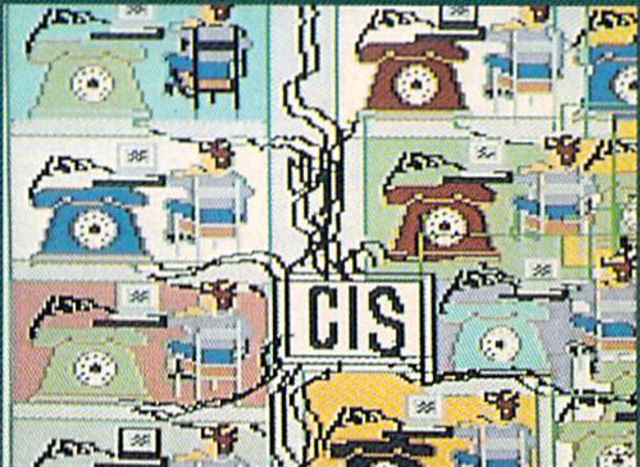
I hope that you will find *Fastnew* a useful utility. I would like to extend my appreciation to my colleagues at R. J. Brachman Assoc., Inc. for their assistance in developing this program. I welcome correspondence from readers. Write to Don Lewis, P.O. Box 521, Folsom, PA 19033. Please include a stamped and self-addressed envelope if you desire a reply. ☐

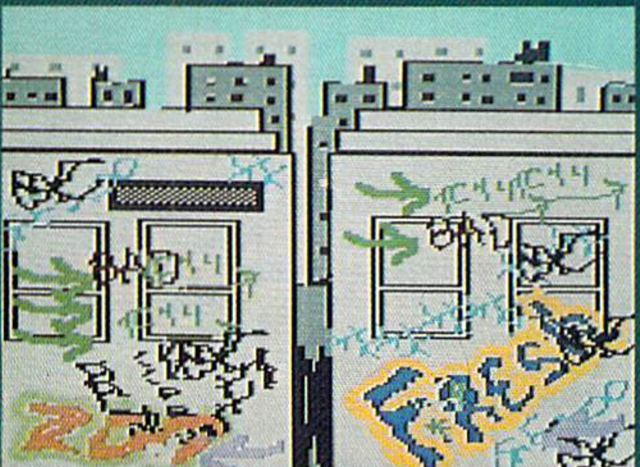
SEE PROGRAM LISTING ON PAGE 111

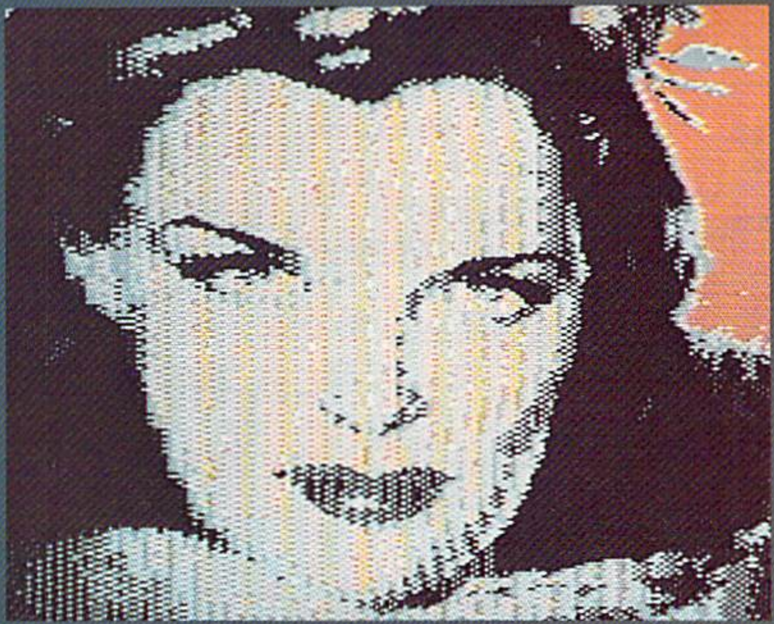
ART GALLERY



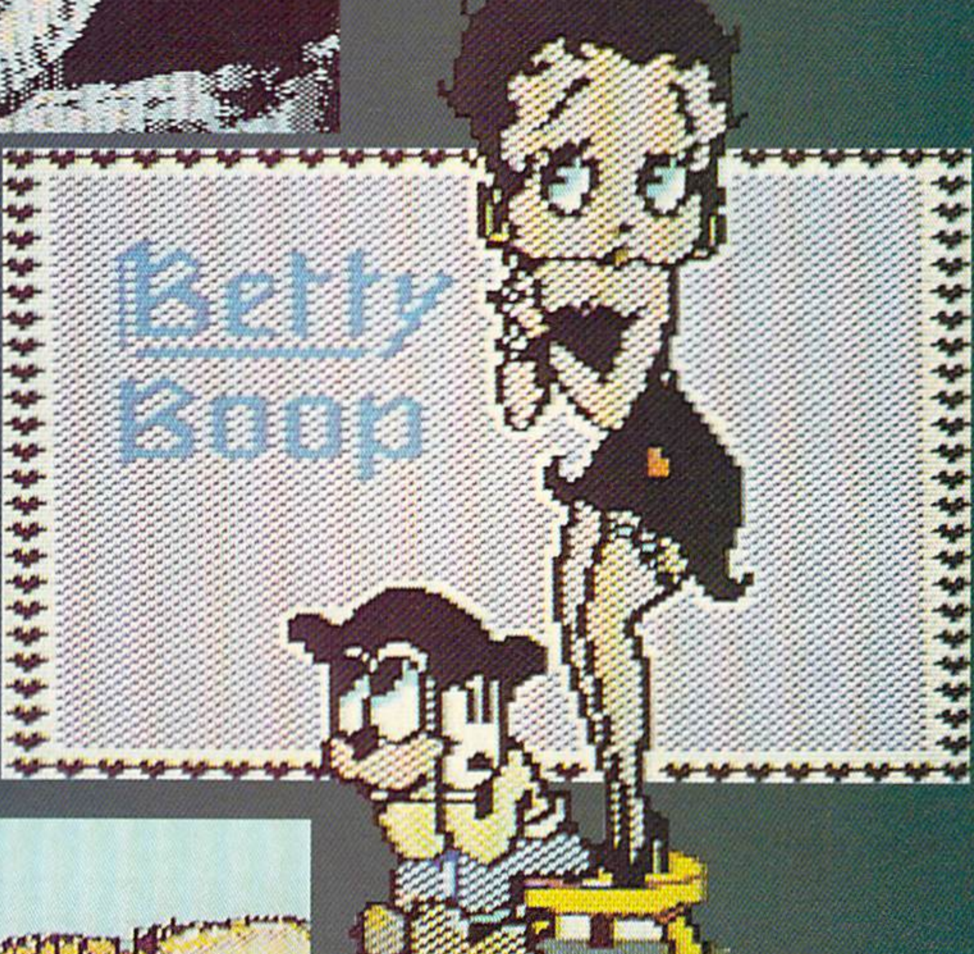
All of Michael Montauck's (Brooklyn, NY) images were created with the Koala Pad and *Koala Painter* software: his representations of *CIS* (the CompuServe Information System) and *Graffiti* (bottom), and the swinging cats at *Ahoy!* (bottom right, page 35).



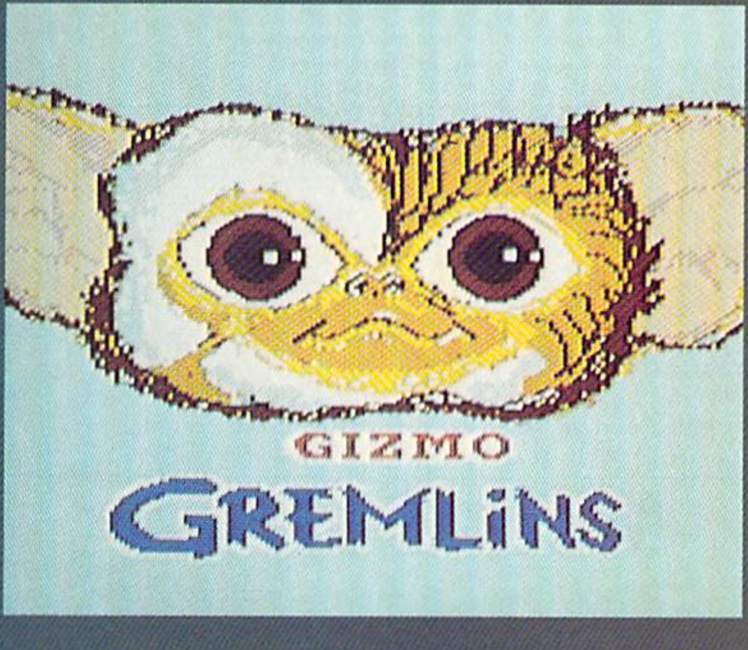


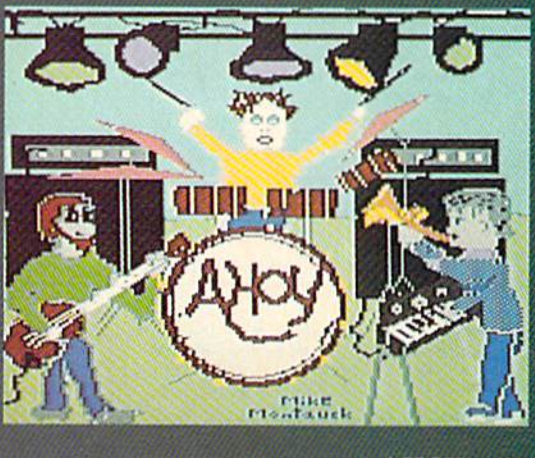


Contributors to *Ahoy!*'s Art Gallery will receive royalties based on the sale of disks containing the best computer graphics received, both published and unpublished. Send your best work on disk, accompanied by a stamped and self-addressed mailer, to Morton Kevelson, P.O. Box 260, Homecrest Station, Brooklyn, NY 11229. Indicate the drawing package that was used to create the image. If you employed a bit map of your own design, indicate the appropriate file parameters, i.e., hi-res or multicolor, location of bit map, screen or color data.



Responding to our initial request for starving computer artists, John Matthews Jr. (San Antonio, TX) indicates that he is starving, but only for attention. Hence his choice of attention-getting celebrities for his images, all created on the Koala Pad. *Betty Boop* (right) and charter Gremlin *Gizmo* (below) were drawn freehand, pixel by pixel, using Koala zoom mode. *Yosemite Sam* (left) was drawn freehand, based on a grid designed to proportion the screen pixels to match the proportions of the Okimate 10 color printer. (The proportions actually look better on the printout than on the screen shot shown.) John digitized his image of *Jane Russell* (above) with Computereyes (see review, p. 66) and added finishing details by hand.





MSD Sure Copy

At last a complete utility package for the MSD Dual Drive. **This is the first MSD utility program that does it all.** The main menu options include:

Sure Copy will put all errors automatically on disk: 20, 21, 22, 23, 27 and 29's.

- Copy Protected Disk
- Copy Files
- Format a Disk
- Change Disk Name
- Quit
- Copy Unprotected Disk
- Scratch a File
- Rename a File
- View Directory

$39.95

TOP SECRET STUFF I



- DMS
- ½ Track Reader
- ½ Track Formatter
- Drive Mon
- The Doc
- Sync Maker
- Sync Reader
- Change Drive No.
- Disk Logger
- Disk Match
- New Wedge
- ID Check
- Unscratch
- View RAM
- Read/Write Test
- Repair a Track
- Fast Format

$19.95

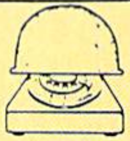
TOP SECRET STUFF II

All C128 Compatible

- Split Screens (Horizontally)
- Smooth Scrolling
- Save Ram From Under Roms
- No Drive Rattle On Errors
- Triple Drive Head Speed
- Autoboot Maker
- Koala Screen Dump
- Display GCR
- Fast Disk Eraser
- Protect Scheme For Your Disks
- Write Protect Disk
- Unwrite Protect Disk
- Mini D.O.S. Wedge
- Fast Diskmatcher
- Data Statement Maker
- Unnew
- 3 Minute Copy
- D.M.S.

$19.95

WAR GAMES AUTODIALER



1-**Auto Dial** will automatically dial a set of numbers you choose.
2-**Review Numbers** will review numbers that were answered by a computer.
3-**Save Numbers** will save numbers where a computer answered.
4-**Hardcopy of Numbers** will print out list of numbers where a computer answered.
5-**LOAD Numbers** will load in numbers to continue where it left off.
6-**Continue** will pick up dialing where it was interrupted.

$29.95

D-CODER

- Translates any machine language program into easy-to-read English descriptions with complete explanations of each command!
- Makes complete notations of all important memory locations accessed by the program! (SID, VIC, MOS, KERNAL, etc.)
- Gives you three ways of accessing programs:
  1) **Will read and list programs from DISK**
  2) **Will read and list programs from MEMORY**
  3) **Direct user input** (from magazines, etc.)
- Can be used to locate and examine any machine language program's protection routines!
- Can be used to easily break apart machine language programs for study and examination!
- Printer option for complete hard copy listings!



You Now Longer need to be an EGGHEAD to read Machine Language.

$19.95

SOFTWARE PROTECTION HANDBOOK

3rd Edition

THIRD EDITION! NOW AVAILABLE!

If you're **tired** of being harassed by **protected software** and too many **copy** programs, then this is the book for you! This 224 page manual covers the gambit from legalities to protection methods to step-by-step back up procedures. Now you can learn both how to **protect and unprotect** software! The techniques covered include copying cartridges to tape or disk, tape protection, and disk protection. Disk protection covers error no.'s 20, 21, 22, 23, 27 and 29 **plus** single track formatting, header modification, header swapping, half track reading and writing, reading and writing modified bit densities, formatting illegal tracks/sectors, sync writing and **more!** The Third edition **explains**, tells how to **detect** and how to **write** them with included software. Eleven useful utilities and many protection listings! Our disk analysis programs reveal the protection methods used on your originals. A diskette with all software is available for a minimum extra charge. This may not be the only book your should have for the C-64, but it is certainly **the one** book you should not be **without!**

C64 Book only	$19.95 US
Book & Disk of all programs	$29.95 US
Vic 20 book . . . Cart. & Tapes only	$9.95 US

THIS MANUAL DOES NOT CONDONE PIRACY

*SHIPPING: $2.00

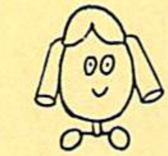
N-CODER

the machine language manipulator...

The perfect companion program to D-CODER!

Allows you to easily make changes in machine language programs... right on the disk!

- Rewrite ability allows code to be altered and then rewritten directly to the disk!
- Features sector-by-sector scrolling assembly language display of machine language programs!
- Notation of ASCII text equivalents for easy spotting of embedded text strings!
- handy reference display of all assembly language commands and their ML numerical equivelents!
- Byte splitter for easy splitting of decimal addresses into low byte-high byte format!

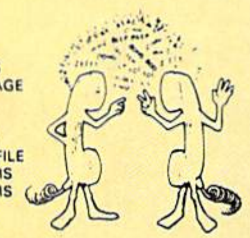


$19.95

Bulletin Board

Set up and operate your own bulletin board with one or two disk drives. This one has all the features and you can customize it easily yourself.

1-RUN MEGASOFT-BBS
2-CREATE MEGA FILES
3-ADD TO SYSOP'S CORNER
4-NEW SYSOP'S CORNER
5-READ MESSAGES
6-SCRATCH MESSAGE
7-CYCLE MESSAGES
8-READ SYSOP MESSAGES
9—WRITE OPENING MESSAGE
10-READ LOG
11-CYCLE LOG
12-READ DOWNLOAD FILE
13-SCRATCH DOWNLOAD FILE
14-ADD TO OTHER SYSTEMS
15-CREATE OTHER SYSTEMS



$59.95 **BBS**

- Leveled Access
- Private Message Base
- Up to 300 Passwords
- Expert Mode
- Open Chalk Board for Highest Level Access

Enclose Cashiers Check, Money Order, or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. Canada orders must be in U.S. Dollars. VISA — MASTER CARD — C.O.D.

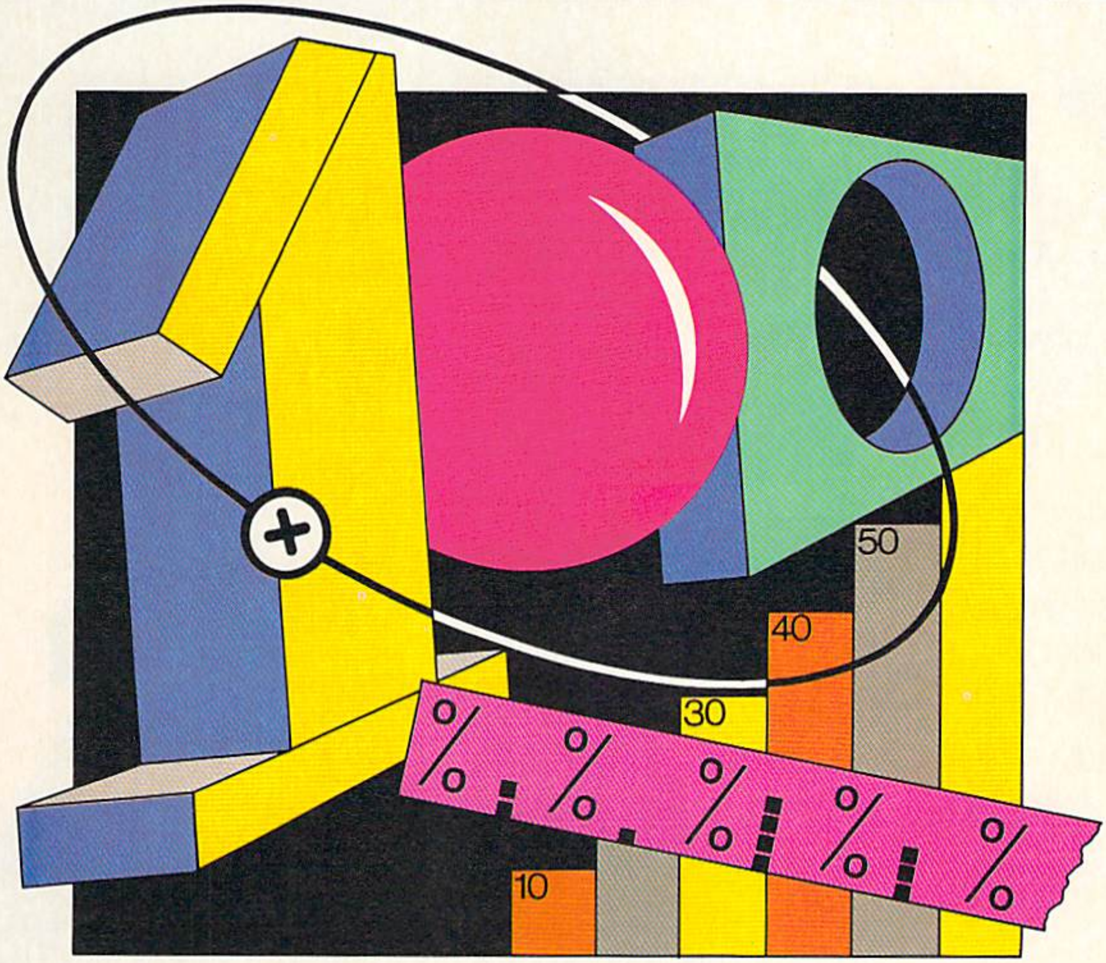
Programs for C-64 $2.00 S & H on all orders

Software Submissions Invited.

MegaSoft Limited

P.O. Box 1080, Battle Ground, Washington 98604

Phone (206) 687-5116 • BBS 687-5205 After Hours Computer to Computer



Tumbling Dice & Data Structures

BY DALE RUPERT

The next roll of the dice may determine your future. Three dice rattle in the cup. The question is this: What is the probability that you will roll an eleven? You confidently reach for your portable computer and quickly load the three-dice simulation program. Within moments it has calculated the answer to this final question on your statistics exam.

This article will not reveal the answer to the question. We will develop some programs to solve it, though. We will also look at a loftier topic than dice rolling, called data structure. The data structures of a program affect its speed, versatility, and debuggability. We will look at some of the tradeoffs of using different data structures.

As we saw last month, there are essentially three ways to determine probabilities: (1) formula evaluation, (2) random simulation, and (3) enumeration. It is very unlikely that any reference book contains the formulas of specific outcomes for the roll of three dice, so Method 1 is no good for solving our problem. This month we will develop procedures for implementing Methods 2 and 3.

Method 2 is sometimes referred to as a Monte Carlo approach. It is often used when the total number of possible outcomes is so large that it is impossible to list them all. Method 3 lists all possible outcomes and is feasible only in cases where that number is small enough and determinable.


SHADOW
THE SHADOW KNOWS
99.9%
EFFECTIVE
Shadow is a new and revolutionary way to duplicate even your most protected software. It encompases all the latest advances in software, as well as a highly sophisticated piece of hardware. This is absolutely **the best** utility available today. "It will even copy the other copy programs." Because of Shadow's unique abilities, we feel DOS protection is a thing of the past.
By the time you place your order we expect the Shadow to copy 100% — that's right, 100% — of **all** software available for the C-64.
Order by phone 24 hrs./7 days or send cashier's check/money order payable to Megasoft. Visa, MasterCard include card # and exp. date. Add $3.50 shipping/handling for continental U.S., $5.50 for UPS air. CODs add $7.50. Canada add $10.00 Other foreign orders add $15.00 and remit certified U.S. funds only. Distributors invited and supported.
MegaSoft LTD.
INTRODUCTORY OFFER
$89.95
P.O. Box 1080 Battle Ground, Washington 98604
Phone (206) 687-5116 • BBS 687-5205 After-Hours Computer-to-Computer Ordering
Reader Service No. 149

It is easy to calculate all possible outcomes for three dice. These nested FOR-NEXT loops generate every combination:

```
10 FOR A=1 TO 6 : FOR B=1 TO 6 : FOR C=1
 TO 6
20 PRINT A;B;C
30 NEXT C,B,A
```

Instantly the most tedious part of statistics class is eliminated! You might send this output to the printer. Then you could find the sum of each triple and keep track of how many of those totals equal the desired value. The probability of rolling an eleven is the quotient of the number of "successful outcomes" divided by the number of possible outcomes. Here a successful outcome is one in which the sum of the three dice is eleven.

Of course we will let the computer keep track of the sums and any other information about the dice that we desire. But before we proceed with this enumeration process, let's try the random simulation method.

ON TO MONTE CARLO

Very little work is needed to simulate and analyze the rolling of three dice, as the *Dice Simulator* program on page 99 shows. Random values from 1 to 6 are chosen for each of the dice. Their total value is calculated and compared with the desired value. If their sum is eleven, the variable GOOD is incremented. The variable N keeps track of the total number of rolls. The number of "good" rolls divided by the total number of rolls should approach the probability we are looking for. Line 70 displays 100 times this probability, giving us the percent probability.

Supposedly the longer this program runs, the closer the result comes to the theoretical value. Of course it is possible that the first three dice we roll have a sum of eleven, giving a 100% probability. On the other hand, we may roll 100 sets of dice before getting a sum of eleven. Neither result is very accurate. Only after numerous rolls with a good random number generator will the results settle down to a value close to the true value.

We have seen in previous articles that the random number generator in our microcomputers is not truly random. The non-randomness may cause the program to "home in" on an inaccurate value. Using an imperfect random number generator is equivalent to rolling loaded dice. I suggest that you run the *Dice Simulator* program to see if you can determine the actual probability before you evaluate it exactly in our next program.

The disadvantages of using this Monte Carlo method should be clear. It may require a fair amount of time to arrive at a reasonable value. Also, it may be difficult to know how reasonable the value is. If the random number generator is not very good, the answer may not be very good either.

The main advantage of this method is that you don't have to be able to list all possible outcomes or even know how many there are. For simulating radioactive decays and molecular motion (topics for next month), this is certainly a desirable characteristic.

You can easily modify the *Dice Simulator* program to answer other questions, such as:

1) Are you more likely to roll a seven or an eleven with three dice?

2) What is the probability of rolling a total value greater than fifteen?

3) What is the probability that the value on one die equals the sum of the values on the other two dice?

4) Is there a fifty percent probability that all three dice show an even number?

Use your intuition to make a guess on each of these before running your program. Probabilities are not always intuitive.

To help you get started on Problem 1, you might add this line:

```
65 IF TTL=7 THEN G2=G2+1
```

Also change line 70 to print the value of 100*G2/N next to the other probability for a side-by-side comparison.

MORE SOPHISTICATION

The bare-bones *Dice Simulator* program uses the most basic data structure, simple numerical variables. The next program goes to the other extreme. It contains numeri-

cal arrays which are indexed by other numerical arrays.

When three dice are rolled, there are 216 possible outcomes. Each die may have any one of six possible values, and 6*6*6 is 216. The computer can readily handle this many items. The program *Dice Analyzer* (on page 99) generates all of the 216 outcomes and stores the values of each of the dice in a numeric array. This program analyzes the probabilities of various events by the exact process of enumeration. Once the list of outcomes is determined, the computer may look through the list, identifying "successful" outcomes and calculating the desired probabilities. A successful outcome is one which meets certain criteria, such as the sum of the three dice being eleven.

The value of each die for every one of the 216 possible outcomes is stored in the numeric array ROLL(N,D) where N is the outcome number (1 through 216) and D is the die's identifying number (1 through 3). If ROLL(7,2) equals 4, then the value of the second die is four on the seventh roll of the dice. If the third die had a value of five on the twentieth roll, then ROLL(20,3) equals 5.

In addition to keeping track of each die's value, the program stores the count of each face value for every roll in the numeric array CT(N,V). Again N is the outcome number, and V is the face value (1 through 6). If CT(9,2)=1, then only one die had a value of two in the ninth roll. If no dice had a value of two in the 216th roll, then CT(216,2)=0.

One other numeric array is filled as the various outcomes are enumerated. The array TTL(N) stores the sum of the three dice on the Nth roll. If TTL(40)=7, then a total value of seven occurred in the fortieth roll.

Lines 40 through 110 enumerate the outcomes and fill the arrays. This program determines exactly the probabilities that

1) Three ones are rolled.
2) At least one three and no sixes are rolled.
3) No twos are rolled.
4) The sum of the three dice is eleven.

This program is actually "overkill" for the sake of illustration. It is certainly not necessary to set up such a complicated set of arrays in order to find these probabilities. We could keep a tally of all successful outcomes as the values are generated in line 60. In fact, the simpler method would run more quickly and use less memory.

Why bother with this more complicated program? With the more sophisticated data structure we gain greater ease of analysis and greater flexibility in displaying the results. To implement Test 2 listed above with the simpler structure, we might have to use line 65 such as:

```
65 IF (A=3 OR B=3 OR C=3) AND (A<>6 AND
B<>6 AND C<>6) THEN T2=T2+1
```

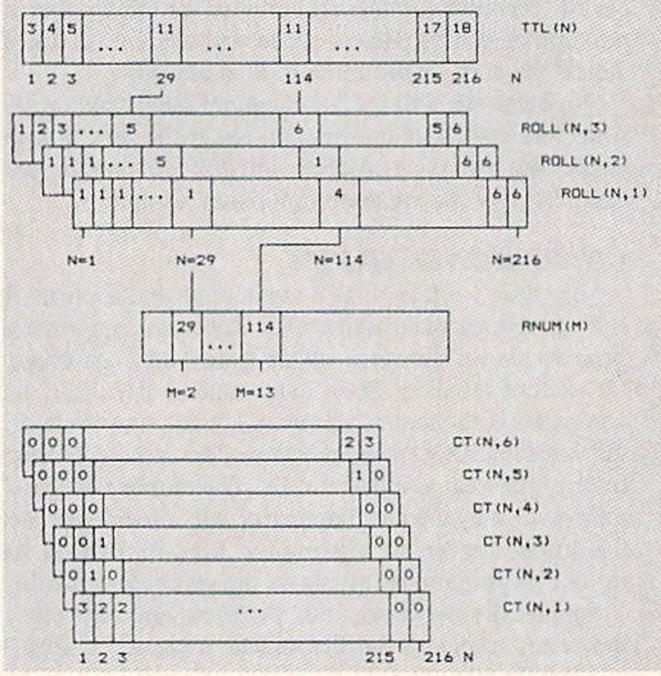
This is somewhat more cumbersome than the corresponding statement in line 150:

```
150 IF CT(N,3)>=1 AND CT(N,6)=0 THEN T2=
T2+1
```

Try to determine the probability that exactly one three and no sixes are rolled. I don't think such a test could be contained in one line with the simpler data structure. (Let me know if I'm wrong!) Line 150 could be easily modified by removing the "<" to perform such a test.

The ROLL array is needed only if selected outcomes are to be listed, as was done for Test 4. RNUM(M) keeps a list of all the roll number indices which satisfy Test 4. For example, if the twentieth roll had a sum of eleven, and if four previous rolls also had sums of eleven, RNUM(5) would equal twenty. RNUM is used as an index into ROLL so that the events satisfying Test 4 can be printed in line 290.

DATA STRUCTURE OF *DICE ANALYZER*



The diagram shows the relation between TTL( ), ROLL( ), and RNUM( ). On the 29th roll (N=29), the dice had values 1, 5, and 5. The sum of their values is eleven so TTL(29) equals 11. This was the second occurrence of a total value of eleven. Consequently RNUM(2) equals 29. Similarly the 114th roll was the thirteenth occurrence of an eleven. The diagram shows that RNUM(13) equals 114, TTL(114) equals 11, and the three dice had values 4, 1, and 6.

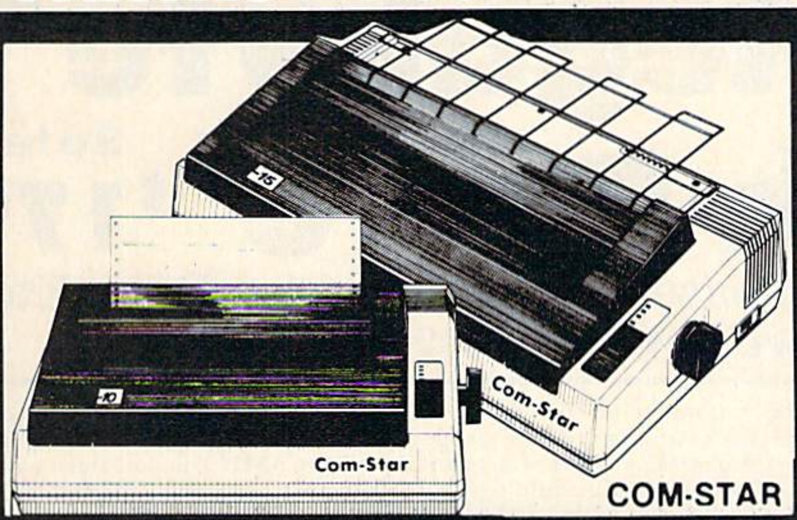
We have covered the fundamental concepts of indexing, pointers, and linked lists in the *Dice Analyzer* program. Study the program and the data structure to understand how the variables are defined and related. Better yet, see if you can modify the program to handle more dice. Perhaps you could change the *Dice Analyzer* program to answer the four questions suggested earlier for the *Dice Simulator* program.

Think about the possible data structures as you begin to create any program. A more complicated structure might be harder to implement, but it may pay off with ease of expandability and versatility later. ☐

SEE PROGRAM LISTINGS ON PAGE 99

Reader Service No. 147

FANTASTIC COMPUTER PRINTER SALE!!!



10X COM-STAR**
HIGH SPEED
Tractor Friction Printer
130-150 CPS

Only **$199**

List $499

15½" 150-170 CPS Printer $319.00

- *Lowest Sale Price, Premium Quality, Tractor/Friction Printer in the U.S.A. (Best Value)*
- *High Speed 130-150 Characters Per Second* • *40, 46, 66, 80, 96, 132 line spacing*
- *Word Processing, Letters* • *Business Forms* • *Labels, Graphics, Tables* • *List Programs*
- *Fantastic Graphics* • *Print Modem Data* • *The Most Important Accessory For Your Computer*

Premium Quality 130-150 CPS 10X COM-STAR High Speed Printer $199

10" carriage, 2K buffer, prints 8½"x11" standard single sheet or continuous feed paper, Bi-directional, impact, dot matrix, 130-150 CPS, 9 x 9 dot matrix with double strike capability for 18 x 18 dot matrix (near letter quality), high resolution bit image, underlining, downloadable characters, true lower descenders with super and subscripts, prints standard, block graphics, and special characters. It gives you print quality and features found on printers costing twice as much!! (Centronics Parallel Interface) (Better than Epson FX80 and Gemini 10x).

List $499.00. Sale $199.00.

Premium Quality 150-170 CPS 15½X COM-STAR Business Super High Speed Printer $319.00

Has all the features of the 10X COM-STAR PRINTER plus 15½" carriage and more powerful electronics components to handle large ledger business forms! (Better than Epson FX 100 & Delta 15).

List $599. Sale $319.00.

SMITH CORONA

10" 80 CPS Printer

Friction/Tractor Bidirectional Dot Matrix printer with six pitches including condensed and enlarged type for impeccable printing. Drop in cassette ribbon; centronics interface included.

List $299.00. *Summer Sale $149.00 (Limited Quantities)*

Super Summer Sale Ends 9-15-85

JUKI®

Printer/Typewriter Combination $229.00

"JUKI" Superb letter quality, daisy wheel printer/typewriter combination. Two machines in one — just a flick of the switch. 12" extra large carriage, typewriter keyboard, automatic margin control and relocate key, drop in cassette ribbon! (90 day warranty) centronics parallel or RS232 serial port built in (Specify). List $349.00. Sale $229.00.

Olympia

Executive Letter Quality $299.00 15" Daisy Wheel Printer

This is the world's finest daisy wheel printer. Fantastic letter quality, up to 20 CPS bi-directional, will handle 14.4" forms width! Has a 256 character print buffer, special print enhancements, built in tractor-feed (Centronics Parallel and RS232C Interface) (90 day warranty).

List $649.00. *Summer Sale $299.00*

Olympia

Printer/Typewriter Combination $399.00

Better than IBM Selectric. Superb computer printer combined with the world's finest electronic typewriter. Two machines in one, just flick the switch for up to 20 CPS printing (**300 Words per minute**) on a 15" carriage that handles up to 14 1/8" in. paper. Drop in cassette ribbon — express lift off correction, Centronics parallel interface (90 day warranty).

List $749.00. *Summer Sale $399.00.*

• *15 Day Free Trial — 1 Year Immediate Replacement Warranty*

PARALLEL INTERFACES

For VIC-20 and COM-64 — $59.00. Apple — $59.00. Atari — $59.00.

Add $14.50 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $29.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars.
WE DO NOT EXPORT TO OTHER COUNTRIES, EXCEPT CANADA.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days delivery. 2 to 7 days for phone orders. 1 day express mail!
VISA — MASTERCARD — C.O.D. No C.O.D. to Canada or APO-FPO

PROTECTO

We Love Our Customers

22292 N. Pepper Rd., Barrington, Illinois 60010

312/382-5244 to order

NEW

Commodore 64

NEW

One Megabyte Disk Drive

Sale **$179**

Sale **$179**

Store Spreadsheets, Databases, Wordprocessing Data, Etc.

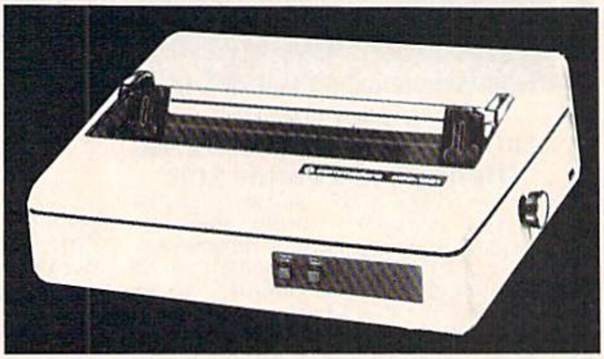
• Commodore 64 • PET • 8032 • B128

The one megabyte disk keeps you from hunting through hundreds of disks for your programs; plus running out of room on your drive for Data base Data, Word processing Text, Spreadsheet Data, Business program Data, etc. With the One Megabyte Disk Drive you can store over 5½ **times the capacity of the 1541**. You can store your own programs and any back-upable commercial programs plus data from your business programs*. Perfect as a second Drive!!!! Enter the world of professional computers today. *C-64 requires IEEE interface.*
List $899.00. **Sale $179.00.** ***LIMITED QUANTITIES!*** *Requires 2 drive software.

$199.00 Sale $199.00

15½" High Speed 150-170 CPS

BUSINESS PRINTER



The 8023 is a highly advanced 136-250 column 15½" professional tractor/friction printer with full Bit image graphics and downloadable characters for custom reports and program listings. The paper feed includes a multiple pin tractor for smooth error free operation. With the ribbon rated at 1 million characters (no mess cartridge) and the print head at 100 million characters this printer will last a lifetime. Full formatting with near letter quality makes lining up decimal points, automatic "$" signs and tabbing look fantastic and easy to use. With out a doubt **THIS IS THE BEST PRINTER VALUE IN THE U.S.A.** List $899.00 **Sale $199.00**. ***LIMITED QUANTITIES!***

SPECIFICATIONS

PRINTING METHOD
Serial Impact Dot Matrix

PRINT RATE
150-170 Characters per second (CPS)

PRINT STYLE
Near Letter Quality

PRINT DIRECTION
Bi-directional

COLUMN CAPACITY
136 - 250

LINE SPACING
Programmable

COPIES
3, including original

RIBBON TYPE
Cartridge ($14.95)

RIBBON LIFE
1 Million Characters

PAPER WIDTH
3" to 15½" tractor or single sheet friction

INTERFACE
IEEE Protocol

CHARACTER SIZE
0.116" high, 0.08" wide

GRAPHICS
Bit Image
Programmable Characters
Reverse Characters

ERROR HANDLING
Internal
Self - Diagnostics
Microprocessor

Commodore 64 IEEE Interface

This interface plugs into your Commodore 64 Disk Drive port and allows you to hook up the 1 Megabyte Disk Drive and 8023 Printer as well as other IEEE devices. Separte power supply insures reliability. Fantastic Interface.
List $109.95. **If bought with printer or disk drive $69.00.**

(no interface needed for PET, B-128, and 8032 computers)

Add $17.50 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $35.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!
VISA — MASTER CARD — C.O.D.
No C.O.D. to Canada, APO-FPO.

PROTECTO
We Love Our Customers
22292 N. Pepper Rd., Barrington, Illinois 60010
312/382-5244 to order

Reader Service No. 147

COLOR MONITOR

Sale SALE!!!

16 COLORS



(Premium Quality)

- **Built in Speaker and Audio**
- **Front Panel Controls**
- **For Video Recorders**
- **For Small Business Computers**
- **Apple - Commodore - Atari - Franklin - etc.**

Add $14.50 shipping, handling and insurance. Illinois residents please add 6% tax. We do not ship to foreign countries (including Canada), Alaska, Puerto Rico, APO-FPO. Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. 1 day express mail!

13" Color Computer Monitor

*Connecting cable $9.95

(Premium Quality)

- **Beautiful Color Contrast**
- **High Resolution**
- **Separated Video**
- **Sharp Clear Text**
- **Anti-Glare Screen**
- **40 Columns x 24 Lines**
- **Supports 80 Columns**

List $329.00

Reg. Sale ~~$179.00~~*

Summer Sale $169.95*

14" COMMODORE 1702 COLOR MONITOR

(add $14.50 shipping) List $299.00 **Super Summer Sizzler Sale $179.00***

15 Day Free Trial - 90 Day Immediate Replacement Warranty

12" XTRON SUPER HI-RESOLUTION MONITOR

80 Columns x 24 Lines, Super Hi-Resolution 1000 lines Green or Amber Super-Clear "Easy to Read" text with special anti-glare screen! *(Ltd. Qty.)* List $249.00 **Super Summer Sizzler Sale $109.95***

12" ZENITH HI-RESOLUTION GREEN OR AMBER TEXT DISPLAY MONITOR

80 Columns x 24 Lines, Hi-Resolution, crisp clear easy to read text with anti-glare screen! A MUST for word processing. *(Ltd. Qty.)* List $199.00 **Super Summer Sizzler Sale $89.95***

12" MONITOR GREEN OR AMBER TEXT

80 Columns x 24 Lines, easy to read up front controls *(Ltd. Qty.)* List $159.00 **Super Summer Sizzler Sale $69.95***

13" RGB & COMPOSITE MONITOR

Must be used to get 80 columns with C128 & Atari ST. List $399.00 **Super Summer Sizzler Sale $279.00***

* Connecting cable $9.95

• LOWEST PRICES • 15 DAY FREE TRIAL • 90 DAY FREE REPLACEMENT WARRANTY
• BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL • OVER 500 PROGRAMS • FREE CATALOGS

Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA. Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders, 1 day express mail!
VISA — MASTER CARD — C.O.D. No C.O.D. to Canada, APO-FPO

PROTECTO

We Love Our Customers

22292 N. Pepper Rd., Barrington, Illinois 60010

312/382-5244 to order



"Our operator will show you how easy it is to operate. At the moment, however, he's getting his diaper changed..."

ERRATA

We omitted two characters from line 620 in *Lucky Lottery* (July '85), causing the program to crap out with a syntax error. Correct line 620 to read as follows:

```
·620 IF A$<>"[F1]" OR A$<>"[F3]" OR A$<>"
[F5]" THEN580
```

In *Duck Shoot* (June '85), a printer malfunction added extra spaces to lines 10 and 120-160. These spaces should be ignored.

We apologize for any problems caused by the above errors. Remember that corrections to *Ahoy!* programs are posted on our bulletin board (718-383-8909 – modem required!) as soon as errors are spotted.

SMALL THINGS CONSIDERED

Ahoy! offers its congratulations to

Kenny Hill
Jamaica, NY

winner of a subscription to *Ahoy!* courtesy of New York's *Small Things Considered* radio show (heard weeknights 5-8 and Saturdays 6-8 on WNYC AM83).

PROGRAMMING TIPS WANTED

Here's a chance for you, the readers of *Ahoy!*, and us, the editors of *Ahoy!*, to work together. If we each do our part, we can make *Ahoy!*'s soon-to-debut programming tips column the finest in any magazine. Your part will be to provide hints, tips, and programming routines that can simplify your fellow Commodore computerists' lives, enhance their abilities, or both. Our part will be to come up with a catchier title by the time this column actually appears.

Keep run-of-the-mill, twice-told tips to yourselves. This is *Ahoy!* We want valuable programming tips. And we'll pay premium rates for them.

Send your submissions to *Ahoy! Programming Tips*, Ion International Inc., 45 West 34th Street—Suite 407, New York, NY 10001.

A Look Around the 1571

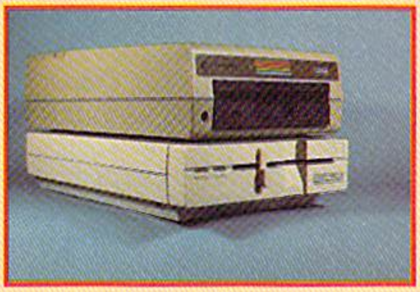
Commodore's Disk Drive Companion to the 128

After taking a close look at the capabilities of our 1571 prototype, we have been very favorably impressed. In most areas operating speed has been improved, as much as nine times for a straight LOAD. Even faster performance can be obtained under certain conditions. Commodore also appears to have learned their lessons on the 1541 very well, as careful attention has been given to previously troublesome hardware details.

SPECIFICATIONS

To fulfill its primary purpose of supporting the Commodore 128 computer, the 1571 has two distinct personalities. As a Commodore drive it behaves very much like a double sided 1541 disk drive. Formatting a disk (a 43 second process) results in tracks 1 to 35 being laid down on side 0, the same as on the 1541. The flip side, side 1, contains tracks 36 to 70. Note that side 0, the side which the 1541 uses, is the underside of the disk when it is placed on a surface with the label facing up. As expected, double siding the disk also doubles the overall data storage capability. However, not all of the drive capacities are doubled.

Total disk capacity for a Commodore formatted disk is now up to 349,696 bytes. A look at the directory of a disk freshly formatted on a 1571 shows a total of 1328 blocks free. The maximum size of a sequential file has been increased to a total of 337,312 bytes. A file this size will





How the 1541 and 1571 drives stack up.
READER SERVICE NO. 101

By Morton Kevelson

span both sides of the disk. Interestingly enough, the maximum size of a relative file is 167,132 bytes, the same as on the 1541.

Track 18 on side 0 is reserved for the block availability map (BAM) and the disk directory. The BAM tells the disk drive which sectors have been reserved for data storage. Sector zero of this track contains the BAM for both sides of the disk. A separate copy of the BAM for side 1 is maintained in sector 0 of track 53. The capacity of the directory is limited to 144 file entries, as on the 1541, although the remainder of track 53 does not appear to be used for any purpose.

THE DOS

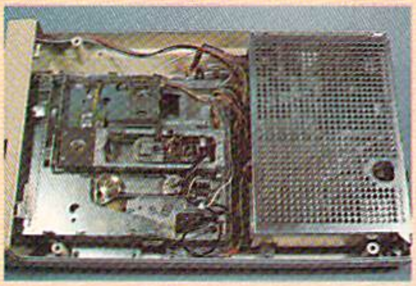
Much work appears to have gone into the built-in disk operating system (DOS), so much so that Commodore has upgraded the DOS version to number 3.0. This is puzzling, as DOS 3.0 is also used in the Commodore D9060 and D9090 hard disk units. The most recent DOS for a Commodore floppy disk unit was version 2.7 on the 8250 dual drive and its SFD 1001 single drive counterpart. The latest revision on the 1541 is DOS 2.6.

TIMING

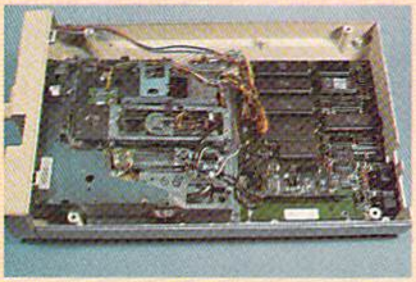
With regard to operating speed we have some good news. As mentioned above, formatting time for both sides of the disk is only 43 seconds. Program loading time, when used with a C-128, has been speeded up by a factor of nine as compared to a C-64 with a 1541 disk drive. The following table shows the result of some simple benchmarks tests:

	C-64/1541	C-128/1571
30K LOAD	77 Sec.	8.5 Sec.
30K SAVE	87 Sec.	58 Sec.
40K SEQ File		
Read & Write	230 Sec.	145 Sec.
Read Only	112 Sec.	45 Sec.
Write Only	120 Sec.	100 Sec.
REL File	271 Sec.	280 Sec.

The sequential file tests were conducted by running the following BASIC program:



The 1571 drive with its lid removed. The power supply is on the right.



Main circuit board rests below power supply and drive mechanism.

```
10 REM DISK DRIVE TIMING
20 REM SEQUENTIAL FILE BEN
CHMARK
30 A$="ABCDEFGHIJK"
40 FORI=1TO8
50 B$=B$+A$
60 NEXT
70 OPEN8,8,8,"SEQ FILE TES
T,S,W"
80 FORI=1TO460
90 PRINT#8,B$
100 NEXTI
110 CLOSE8
120 OPEN8,8,8,"SEQ FILE TE
ST,S,R"
130 FORI=1TO460
140 INPUT#8,B$
150 NEXTI
160 CLOSE8
```

The relative file test consisted of creating a file with 512 records where each record was 254 bytes. Only the last record was written to.

These timing tests are only part of the story. The 1571 has a "burst" mode which should result in much faster data transfer rates between the disk drive and a C-128. The burst mode is recommended only for non-critical data, such as bit maps, where a data error is not fatal. The high speeds used by the burst mode leave little time for DOS error checking routines. Commodore 64 users will not be able to take advantage of the faster operating speeds without special software. The C-64 operating system does not have the required routines. Using a 1571 with a C-64 will not give any speed advantage, although the extra disk capacity will still be available.

The discussion until this point has centered about the Group Code Recording (GCR) scheme which is virtually unique to Commodore formatted disks. The 1571's second personality gives it the ability to format disks using modified frequency modulation (MFM) double density recording techniques. This format has been adopted by a large segment of the personal computer community, most notably by many CP/M machines. Even in this case there are variations among machines, most notably in the total capacity of a formatted disk. This can be related directly to the number of bytes in a disk sector.

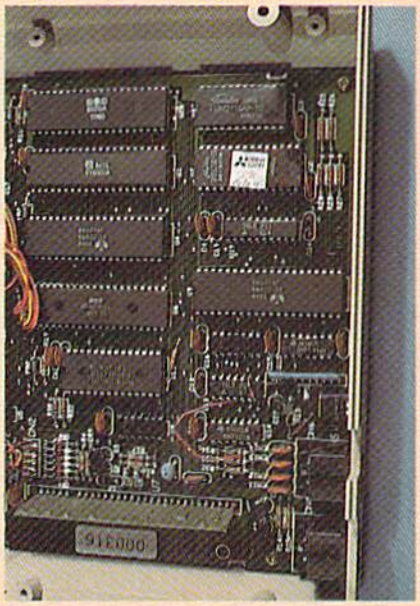
The 1571 has considerable flexibility with the MFM format. Sector capacity can vary from 128 bytes up to 1024 bytes. The following table shows the possible combinations.

Sector Capacity	# of Sectors per Track	Disk Capacity*
128 Bytes	26	260K
256 Bytes	16	320K
512 Bytes	9	360K
1024 Bytes	5	400K

*Formatted capacity for a double sided disk.

The MFM formats, as well as the high speed burst mode, are accessed via a new direct disk command, U0. The preliminary documentation indicated that mixing of the various MFM and GCR formats was possible on a single disk. Even formatting of single tracks is possible. These prospects are sure to gladden the hearts of many copy protection enthusiasts.

We will report on CP/M on the C-128 with the 1571 in the near future. As of this writing (mid-June) Commodore has not released the CP/M operating system and support



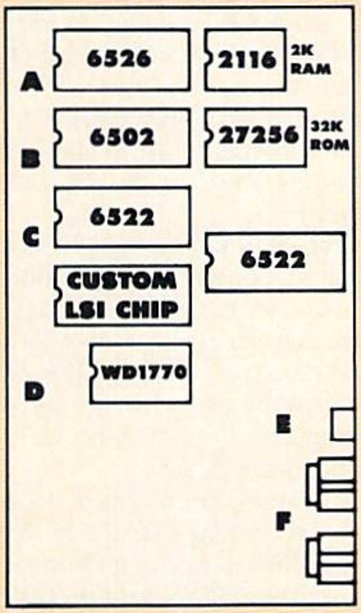
Closeup of the main circuit board. See key at right for chip layout.

utility disks.

COMPATIBILITY

Although the 1571 is a double sided drive, it is still fully capable of reading and writing single sided disks formatted on a 1541 disk drive. When working within C-128 mode, the drive initially assumes that the disk is double sided. If a single sided disk is inserted it takes the drive about 15 seconds to convince itself that the second side is blank. Once done, all subsequent reads and writes are done in single sided mode. Unfortunately this second side checking routine occurs every time disks are swapped. The 15 second delay could become annoying if single sided format disks are frequently mixed with double sided disks.

Double sided disks are filled on side 0 first. If you use a 1541 with a double sided disk you will not be able to access the second side. Flipping the disk will not work as the rotation is reversed. In addition, the track numbers run from 36 to 70. These are illegal as far as the 1541 DOS is concerned. The real difficulty is when a 1571 is added to an existing C-64 system. As the disk fills, files will spill over to the second side. Trying to read one of these on a 1541 will



1571 MAIN CIRCUIT BOARD LAYOUT

A–CIA CHIP
B–MICROPROCESSOR
C–VIA CHIP
D–MFM DISK DRIVE CONTROLLER
E–DEVICE NUMBER SELECTION SWITCH
F–SERIAL PORT CONNECTORS

return an illegal track or sector error. The 1541 does not recognize track numbers greater than 35.

When the 1571 is connected to a C-64 it reverts to 1541 mode. Under these conditions the drive will not read the second side of a double sided disk. Even formatting time reverts to 70 seconds for a single side! Compatibility with 1541 software is very good. All the standard DOS commands function as before. Nearly all of the copy protected disks we tried LOADed without any difficulty.

THE ELECTRONICS

The internal works consist of three modules, two for the drive electronics and one for the hardware. The power supply module is completely contained in a perforated metal cage located directly above the main circuit board. This mounting position insures that heat dissipation will be away from the main electronics. The major heat generating components are the twin voltage regulators and transformer on the left side. These supply the five volt DC and twelve volt DC requirements for the drive. The 120 volt supply line is fully filtered.

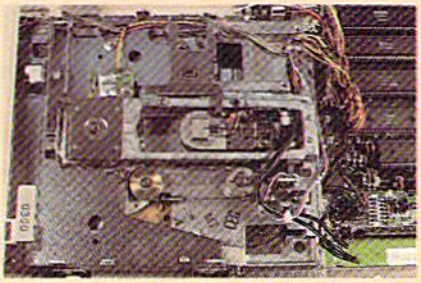
The heart of the drive electronics is a 6502A microprocessor with 32 kilobytes of disk operating system in ROM supported by two kilobytes of RAM. A 6526 CIA chip and a 6522 VIA chip handle the I/O requirements. The Commodore format GCR recording is handled by a custom large scale integrated circuit. The MFM recording for CP/M format disks is taken care of by a Western Digital WD1770 chip.

MECHANICALS

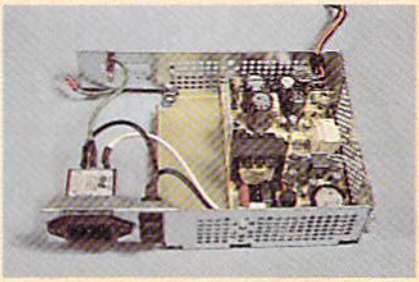
As with all of Commodore's disk drives, the design of the 1571 is revolutionary. For the end user the most significant improvement is the incorporation of an optical sensor for the track one head position. This should virtually eliminate the alignment problems which plagued the 1540/1541 disk drives. The mechanical stops are still present but they are not normally used. The disk operating system arrests head movement at the track one position when formatting a disk or error checking as might be done with some forms of copy protection.

To insure proper seating of the disk, the 1571 spins the drive spindle every time a disk is inserted or removed. This operation is triggered by the write protect sensor. Of course, the same electronics also inform the DOS whenever a disk is swapped.

The 1571 is a double sided drive. Two heads (which in this case are definitely better than one) are provided to accommodate the storage of data on both sides of the floppy surface. These are situated in diametrically opposing positions so as to sandwich the floppy medium between them. These hard, highly polished, precision surfaces must never make contact lest their finish be marred. A mechanical interlock has been incorporated into the latch mechanism for this very purpose. The heads will remain in the open position unless a disk is inserted to trigger the interlock. This is true regardless of the position of the external operating lever.



Closeup of drive mechanism, which incorporates a brushless DC motor.



A closeup and inside view of the 1571 disk drive's power supply.

Head cleaning, for those who shun the costly cleaning kits, requires a minimum of disassembly. Simply remove the top cover to gain access to the heads. As can be seen from the photographs, the power supply and the main electronics board are situated behind and below the drive mechanicals. There is no need to remove any of these components to clean the heads. Do not attempt to force the upper head support arm beyond its mechanical stop. Movement is rather restricted as compared to the design on the 1541. A good-quality clean cotton swab, moistened with isopropyl alcohol (91%), does the job. As with all component disassembly, we caution our readers to use their own good judgment. Opening of the disk drive may void the warranty. If you have any doubts on your ability to handle this work, leave it for a qualified service technician.

The drive mechanism is manufactured by Newtronics/Mitsumi Electric Co. Ltd. It incorporates a brushless direct drive DC motor for spinning the disk. The motor control electronics are on a circuit board which surrounds the armature.

As expected, the back panel includes a pair of serial ports to allow daisy chaining of peripherals, the main power socket, and the on/off switch. A significant new addition is a pair of miniature switches right next to the serial port connectors. These allow the selection of device numbers 8 through 11. Interestingly enough, the preliminary documentation still talks about the cutting of internal jumpers to change device numbers.

In addition to the mandatory disk slot, the front panel sports a twist lock door latch and a pair of colored indicating lights. For some reason Commodore has departed from their previous color scheme. In direct opposition to the 1541 the power on light is red and the drive indicator light is green.

THE MANUAL

Judging from the draft copy of the manual, users will be in for a pleasant surprise. Although it is directed to the reader with some knowledge of BASIC, the manual contains a lot more information than the early editions of the 1541 book. Carefully written explanations with numerous detailed examples are sprinkled throughout the text. The draft manual is over sixty pages, with two more chapters yet to come.

Numerous tidbits of information have been included. Many of these were apparently gleaned from the pages of the various Commodore publications. Commodore has even officially named the ignominious asterisk file as a "splat" file. Splat files occur when a file is opened and not properly closed by the DOS. They list in the directory with an asterisk next to their file names. As many a user has woefully found out, they can foul up a disk if not properly treated. Details for their cure are now an official part of the 1571 manual.

CONCLUSION

The 1571 disk drive makes an excellent companion to the C-128. The system provides all of the features Commodore users have been looking for since the introduction of the VIC 20. Much has been done to improve on the marginal performance of the serial bus as implemented on the 1541. (This bus was actually a low cost compromise which allowed the C-64 to appear on the market in a hurry.)

The flexibility of the 1571's operating modes will be put to full use by the CP/M 3.0 Plus operating system. Several popular disk formats will be directly supported or easily accessed by a set of simple keyboard commands. In particular, the supported formats are:

1. Commodore Formats (GCR)

- The existing CP/M 2.2 disk format, which is used by a Commodore 64 with the CP/M cartridge. Disk capacity is limited to 136 kilobytes in this format.
- A new Commodore CP/M format which will utilize the full 360 kilobyte capacity of a double-sided 1571 disk.

2. Non-Commodore Formats (MFM)

- Osborne (single sided only)
- Kaypro
- Epson
- IBM CP/M-86 (single- and double-sided disk capacities from 160 kilobytes to 360 kilobytes)

Additional formats are possible but will require special programming of the 1571 disk drive. In all cases data files may be transferred between the Commodore 128 and the respective machines. Of course, you cannot run IBM programs since CP/M 3.0 runs on an eight bit Z-80 microprocessor while CP/M-86 runs on a 16 bit 8088 microprocessor.

The 1571 is only the first step in high quality peripheral support for the C-128. Work is already well under way on the 1572, a dual drive version of the 1571. It also would not surprise us to see a Commodore 3¼ inch hard disk with a 10 to 20 megabyte capacity for under $500 in the near future. This combination of features with low cost is sure to encourage many existing Commodore users into the upgrade path. We also expect to welcome many new users into the Commodore fold. □

Touring the Commodore 128 Keyboard

Last month we presented an inside view of the C-128. Starting this month we will take a closeup look at some of the specific features which make this new machine a significant improvement over its predecessors.

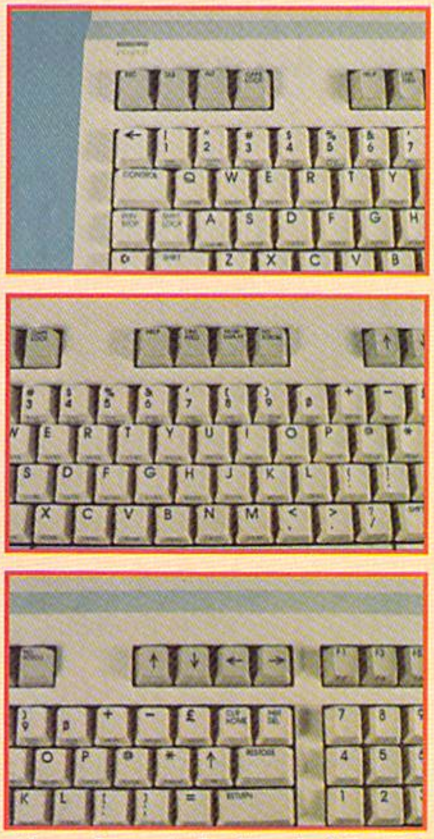
By Morton Kevelson

The ergonomic considerations in the design of the C-128 keyboard are readily apparent. The keyboard's low profile should help reduce fatigue by allowing the user's wrists to rest comfortably on the table. Touch typists will welcome the raised dimples on the F and J home keys, as well as on the 5 key of the numeric keypad. The 2½ inch wide clear space between the keys and the cooling vents is sure to become a haven for all sorts of keyboard overlays. All in all, there are 92 keys to bewilder the novice.

Observant readers will also note that the main part of the keyboard is nearly an exact duplicate of the Commodore 64's. The only notable change is the relocation of the function keys to a horizontal configuration at the right of the top row. Present 64 owners should have very little trouble becoming comfortable with the C-128 layout.

Physical attributes aside, there is a lot more to the C-128 keyboard than meets the eye. Hidden beneath the tactile array, under control of the new operating system, lurks a multitude of functions designed to ease the user's sojourn into the realm of computing with the C-128.

If you have been using a VIC 20 or a C-64, you will already be familiar with a good portion of the keyboard's capabilities. The full screen editor functions in the same way as these machines'. The INST/DEL key still eats characters to the left, or when shifted opens spaces to the right. The quote and insert modes still operate in the same fashion which non-Commodorists find so thoroughly befuddling. The CLR/HOME key still moves the cursor to the top left corner of the screen, or when shifted clears the entire screen. As with the C-64, text can be displayed in 16 colors, as the labels on the 1 to 8 keys indicate. These colors are still accessed by holding down the CONTROL key or the Commodore logo key. Unlike the C-64, the labels on the number keys now indicate both colors which are available for that key. As before, text display can be reversed with a CONTROL/9 and restored with a CONTROL/0.



Above the main keyboard of the 128 are (top) Esc, Tab, Alt, Caps Lock, (middle) Help, Line Feed, 40/80, No Scroll, and (bottom) cursor keys.

The real power of the C-128 keyboard is in the new editing and control features which have been incorporated into the BASIC 7.0 operating system. For convenience, we have classified these features into three categories: the function keys, the ESCape key sequences, and the CONTROL key sequences. Let's take a close look at each in turn.

THE FUNCTION KEYS

On power up, eight useful BASIC phrases are preassigned to the C-128 function keys. These phrases can be viewed by executing BASIC 7.0's new KEY command without any parameters. The following list should appear on the display.

```
KEY 1,"GRAPHIC"
KEY 2,"DLOAD"+CHR$(34)
KEY 3,"DIRECTORY"+CHR$(13)
KEY 4,"SCNCLR"+CHR$(13)
KEY 5,"DSAVE"+CHR$(34)
KEY 6,"RUN"+CHR$(13)
KEY 7,"LIST"+CHR$(13)
KEY 8,"MONITOR"+CHR$(13)
```

ESC	72	7	24	RESTORE	–	N	39
TAB	67	8	27	RUN/STOP	63	M	36
ALT	–	9	32	SHIFT LOCK	–	<	47
CAPS LOCK	–	0	35	A	10	>	44
HELP	64	+	40	S	13	?	55
LINE FEED	75	–	43	D	18	SHIFT (RIGHT)	–
40/80	–	£	48	F	21	CRSR UP/DOWN	7
NO SCROLL	87	CLR/HOME	51	G	26	CRSR LEFT/RIGHT	2
CRSR UP	83	INST/DEL	0	H	29	**KEYPAD**	
CRSR DOWN	84	CTRL	–	J	34	7	70
CRSR LEFT	85	Q	62	K	37	8	65
CRSR RIGHT	86	W	9	L	42	9	78
F1	4	E	14	[	45	+	73
F3	5	R	17	]	50	4	69
F5	6	T	22	=	53	5	66
F7	3	Y	25	RETURN	1	6	77
LEFT ARROW	57	U	30	COMMODORE	–	–	74
1	56	I	33	SHIFT(LEFT)	–	1	71
2	59	O	38	Z	12	2	68
3	8	P	41	X	23	3	79
4	11	@	46	C	20	0	81
5	16	*	49	V	31	.	82
6	19	UP ARROW	54	B	28	ENTER	76

The table at left shows the values generated in location 212 by the keys of the C-128 keyboard. (See **C-128 Keyboard Distinctions** *below for explanation.)*

Getting this list on the screen serves two purposes. First, it reminds you what is programmed into the keys. Secondly, it gives you the opportunity to easily change the contents in any of the eight function keys. For example, if you are not planning to do any graphic programming, you may want to redefine f1. Just cursor up to the "3" in KEY 3 and change it to a "1". Hit RETURN and the directory will now be displayed by f1. Enter

```
KEY 3,"OPEN 4,4:CMD4"+CHR$
(13)
```

followed by

```
KEY 4,"PRINT#4:CLOSE4"+CHR$
(13)
```

and you will be all set for easy printer listing. Note how the numbers 1 through 8 which immediately follow the KEY statement refer only to the function keys. No other keys are affected by this command.

There is one possible disadvantage to having the function keys preprogrammed. BASIC programs written for the C-64 do not expect to find a lengthy message when examining the function keys. This can be easily remedied by reprogramming all eight keys with the following:

```
FOR I=1 TO 8:KEY I,CHR$(I+1
32):NEXT
```

This will place the traditional (for VIC 20 and C-64 users) CHR$ codes into the function keys. You will probably want to do this in every program which uses the function keys to control program operation.

C-128 KEYBOARD DISTINCTIONS

The C-128 sports a numeric keypad which should greatly speed up data entry in numerically intensive applications, most notably spreadsheets. These keys, as well as the discrete cursor keys, duplicate existing keys on the main keyboard. There are applications where a numeric keypad may be of little or no interest. Instead, various program control functions could be assigned to the new keys.

For such applications it would be desirable to distinguish the keypad from the number keys on the main keyboard. The traditional ASCII code check following a GET will not work, since the same ASCII code will be returned by the corresponding keys. For example, pressing a "1" on the keyboard or keypad returns an ASCII value of 49. Fortunately, there is a simple way for BASIC to distinguish which physical key has been pressed.

Locations 211 and 212 are used by the operating system for this purpose. The values stored here every sixtieth of a second are unique for every key. At address 211 the SHIFT, Commodore logo, and CONTROL keys are identified. These keys will set bits zero, one, and two respectively at this address. If you PRINT PEEK(211) a zero will normally be returned. The SHIFT key changes this to a one. The logo key returns a two and the CONTROL key results in a four. These values are additive, so holding all three keys down gives a seven.

The remainder of the keys, with the exception of the RUN/STOP, RESTORE, ALT, and 40/80 keys, place a unique value in address 212 when pressed. On the C-128 this location contains an 88 when no key is pressed. Press any other valid key and a value between 0 and 87 will appear. The table above shows the values generated in location 212 by the keys on the C-128 keyboard. The following program can be used to reproduce these results:

```
10 DO:GETKEY A$
20 PRINT ASC(A$);PEEK(212)
30 LOOP
```

The same program in BASIC 2.0 on the C-64 or VIC 20 would be:

```
10 GET A$:IFA$=""THEN10
20 PRINT ASC(A$);PEEK(203)
30 GOTO 10
```

Note the change in the PEEKed address. The operating systems on all Commodore machines have many similarities. Addresses 203 and 653 on the VIC 20 and the C-64 are equivalent to 212 and 211 on the C-128. The values in 203 for the C-64 will be the same for the corresponding keys in 212 on the C-128. However, the results on the VIC 20 are quite different. The shift flag values are the same for all three computers. You may want to use the KEY statement on the C-128 to clear the function keys before RUNning this program, or strange things may result.

ESCape KEY SEQUENCES

The C-128 has the powerful full screen editor that a veteran Commodore user will not feel at home without. This editor has been greatly enhanced with the addition of a series of ESCape key commands. The result is that the C-128 screen editor approaches the capabilities of a full blown text editor. Writing and editing BASIC programs has never been easier.

There are 21 ESCape key sequences implemented on the C-128, almost one for every letter of the alphabet. These sequences are so useful that you will probably want to become familiar with them in a hurry. The best way to become acquainted with them is just to use them while writing a program. Afterwards, if you really want to see how dependent you have become on the new features, just GO 64 and enter the same program. Be forewarned, you may be in for a frustrating experience.

Each of the ESCape sequence commands are executed by pressing and releasing the ESC key followed by the appropriate letter key. Do not hold down both keys at once. For example, pressing the ESC key followed by the "A" key places the editor in automatic insert mode. Anything you type will automatically push the remainder of the display ahead of it. This is a tremendous convenience for rearranging text between quotes or for adding temporary com-

mands during program debugging. It is especially useful when you keep in mind that the C-128 permits entry of up to 160 characters (four 40 column lines or two 80 column lines) into a BASIC program line. Note that this is not the same as opening a gap by using the INST/DEL key. All of the keyboard control functions, such as the cursor keys and the CLR/HOME key as well as the INST/DEL key, retain their usual functions. In other words, insert mode does not place the screen editor into quote mode.

The screen parser has been enhanced in other ways as well. It will now link up to a full screen of text at one time. By comparison, the C-64 is limited to only two screen lines and the VIC 20 handles four screen lines. However, trying to enter more than 160 screen characters into a single BASIC program line results in a ?STRING TOO LONG ERROR. By comparison, exceeding the C-64's two line limit simply resets the screen linker to the next two lines. This full screen linking on the C-128 must be kept in mind when using the ESCape functions. The erase to end of line command (ESC Q) will remove all text from the current cursor position to the end of screen text or the next carriage return, whichever comes first. The erase to beginning of line command (ESC J) works analogously but in the opposite direction. The same is true of the move to end of line (ESC K) and move to beginning of line (ESC J). On the other hand, the insert a line (ESC I) command is a screen line function. Use of this command within a BASIC line opens up one screen line while maintaining the line links for BASIC editing.

All of these functions can also be obtained under program control. The following sample program illustrates some of the effects which can be easily obtained by using the ESCape sequences in PRINT statements:

```
10 PRINT CHR$(147):REM CLE
AR SCREEN
20 CHAR,14,11,"OFF THE TOP
"
30 CHAR,12,12,"DOWN THE BO
TTOM"
40 CHAR,0,12
50 FOR I=0TO12
60 PRINT CHR$(27)"I"CHR$(2
7)"I";:REM INSERT TWO LINE
S
70 PRINT CHR$(27)"V";:REM
MOVE SCREEN UP ONE LINE
80 NEXT
```

Note that CHR$(27) is the ESCape key. Several keystrokes can be saved by programming PRINT"<reverse [>X" where the reversed left bracket is obtained by simultaneously pressing CONTROL, SHIFT, and the colon keys while in quote mode. The reverse left bracket between quotes in a BASIC statement will generate a CHR$(27). This is a useful trick to keep in mind for sending escape codes to a printer with a PRINT# statement. Note that the ESCape key commands only work when sent to the screen. If you want to see how the program works try adding the following lines:

```
65 SLEEP 1
75 SLEEP 1
```

The SLEEP command pauses program execution for the specified number of seconds. Also note the use of the CHAR command to conveniently position the cursor on the screen. The CHAR command is really meant to display text on a graphic screen. In this context it is being used as a PRINT AT command.

One of the more interesting capabilities of BASIC 7.0 is the ability to define windows. Once a window is set, all subsequent screen output is restricted to this area. Try the following program:

```
10 PRINT CHR$(147)
20 CHAR,10,10,CHR$(27)"T"
30 CHAR,10,10,CHR$(27)"B"
40 LIST
```

This will define a screen window whose upper left hand corner is at the tenth row and tenth column position. The lower right hand corner will be at twentieth row and column position. Note the use of the CHAR statement to position the cursor as well as to set the window coordinates. Also note the coordinates in the CHAR statement of line 30. These are correct as shown. The CHAR statement references its coordinates from the upper left hand corner of the currently active screen window. Since this has been already defined by line 20, we have to take this into consideration for line 30.

To help keep track of the current screen window, BASIC 7.0 includes the RWINDOW function. The following sequence of commands:

```
R=RWINDOW(0)
C=RWINDOW(1)
```

will place the number of rows and columns into the variables R and C.

While entering a BASIC program, the window commands can be used to partition the screen into a message area and a text area. Just write yourself some notes at the top of the screen and set the screen window. Continue to enter BASIC program lines. All screen control functions,

such as clear or home, will not affect the message area. To cancel a window just hit the HOME key twice.

Under program control, a window can be used to eliminate the need to continuously rePRINT a screen layout. The message area in the window can be manipulated as desired without affecting the remainder of the screen. Note that only one window can be active at any time.

A word about the scroll commands. Turning scrolling off, with an ESC M, causes the cursor to wrap from the bottom of the screen to the top of the screen. There is also a NO SCROLL key which according to the preliminary documentation is supposed to perform the same function. This is not so. We found this key to work as a pause as described below.

CONTROL KEY SEQUENCES

Turn on your C-128, hold down the CONTROL key, and press G. Surprise! If you had the volume turned up on your television or monitor, you were rewarded with a pleasant bell tone. This may not seem like much, but if I had a nickel for every time someone asked me for a simple way to make that noise on a C-64, I would have retired a wealthy man some time ago. Commodore has finally implemented the traditional teletype "bell" in the C-128 console. The same tone can be produced with a PRINT CHR$(7).

In general the CONTROL functions are not unique to the C-128. Most of them will be familiar to present users of the VIC 20 or the C-64. The C-128 does have some significant changes and additions to this keystroke sequence. Note that unlike the ESCape key sequence you have to hold down the CONTROL key while pressing the second key. Also, CONTROL key commands can be placed into PRINT statements while the screen editor is in quote mode. In contrast, pressing the ESCape key while in quote mode will activate the ESCape key sequence.

As mentioned above, a CONTROL left bracket will send an ESCape character. Try it. If the editor is not in quote mode, this key sequence will initiate the ESCape mode. This little-known key sequence will also generate a CHR$(27) on the C-64 and the VIC 20. Of course, this has no direct effect on these two machines.

The C-128 has a TAB key on its keyboard right next to the ESCape key. Pressing this key generates a CHR$(9), which moves the cursor to the next tab stop on the screen. The CONTROL I key combination will have the same effect. On the VIC 20 or the C-64, a CHR$(9) will enable the character set switch capability. On the C-128 this has been moved to CONTROL K (CHR$(11)). The disable-character-set-change command has also been changed to CONTROL L (CHR$(12)), although the old CONTROL H (CHR$(8)) has no unique C-128 function. Programmers accustomed to these codes on the VIC 20 or C-64 should take note.

Getting back to the tab functions, the CONTROL X (CHR$(24)) will set or clear a tab at the current cursor position. However, to clear all tabs you have to use the ESCape Z sequence. The best way to learn about the tab stops is just to play with them. Note that a TAB command does not wrap the cursor to the following screen line. Tabbing to the end of a line simply leaves the cursor at that position.

Another change is worth noting. The CONTROL J (CHR$(10)) code generates a linefeed without a carriage return. On the C-64 and VIC 20 both a linefeed and carriage return will be produced.

The remaining unique C-128 CONTROL codes are associated with the 80 column display. CONTROL B (CHR$(2)) turns underline on. However, there is no corresponding CONTROL sequence to turn it off. You will have to PRINT CHR$(130) to cancel underline mode. The 80 column characters can be set to flash by using a CONTROL 0 (CHR$(15)). As with underline mode, there is no CONTROL sequence to turn flash off. Use PRINT CHR$(143).

Continued on page 74

TABLE OF ESCAPE KEY COMMANDS

@	Erase screen from cursor to end of window
A	Automatic insert mode
B	Set bottom right corner of window at cursor
C	Cancel automatic insert mode
D	Delete current linked line
I	Insert a screen line, preserve line links
J	Move to the beginning of current linked line
K	Move to the end of current linked line
L	Turn on scrolling
M	Turn off scrolling
N	Un-reverse 80 column display
O	Cancel insert, quote, reverse, and flash modes
P	Erase from the beginning of current linked line to the cursor
Q	Erase from cursor to the end of current linked line
R	Reverse 80 column display
T	Set top left corner of window at cursor position
V	Scroll screen up one line
W	Scroll screen down one line
X	Toggle between 40 and 80 column display
Y	Restore default TAB stops at every eighth character position
Z	Clear all TAB stops
ESC	Cancel ESCape mode. (Any nonimplemented character will also cancel ESCape mode.)

COMMODARES

PROGRAMMING CHALLENGES

By Dale Rupert

Each month, we'll present several challenges designed to stimulate your synapses and toggle the bits in your cerebral random access memory. We invite you to send your solutions to:

Commodares, c/o *Ahoy!*
P.O. Box 723
Bethel, CT 06801

We will print and discuss the cleverest, simplest, shortest, most interesting and/or most unusual solutions. Be sure to identify the *name* and *number* of the problem you are solving. Also show sample runs if possible, where appropriate. Be sure to tell what makes your solutions unique or interesting, if they are. Programs on diskettes are welcome, but they must be accompanied by listings. You must enclose a stamped, self-addressed envelope if you want any of your materials returned. Solutions received by the middle of the month shown on the magazine cover are most likely to be discussed, but you may send solutions and comments any time. Your original programming problems, suggestions, and ideas are equally welcome. The best ones will become *Commodares*!

Problem #21-1: Geometry Fun

This challenge was suggested by Phil MacLean (Columbus, OH). The user inputs three ordered pairs of numbers (x,y values in Cartesian coordinates), and the computer tells whether or not they represent collinear points. The three points are collinear if they lie on the same line. If they are, the computer tells which lies between the other two on the line. Any values of x and y are allowed. Better dust off the geometry books!

Problem #21-2: Logical Fun

Michael P. Marron (Stony Brook, NY) submitted this *Commodare.* Four students were discussing their upcoming report card marks. They made the following statements:

AL: (1) We'll all get different marks.
(2) If I get a 95, then Connie will get a 65.
BETTY: (3) If Connie gets a 75, then Al will get a 65.
(4) Al will get a better grade than Dan.
CONNIE: (5) If Betty doesn't get a 95, then Al will get a 75.
(6) If I get an 85, then Dan won't get a 65.
DAN: (7) If Connie doesn't get a 95, then I'll get an 85.
(8) If Betty doesn't get an 85, then I won't either.

Your task is to write a program which (A) determines the mark for each student assuming the above statements are all true, and (B) finds all possible solutions assuming statement number (1) is false. For extra credit, determine how many of the original eight conditions can be eliminated and still have only one solution.

Problem #21-3: Decimal Columns

E. Harvey Hammett (Houston, TX) asked for a short method of lining up decimal points when numbers are printed in a column. Your routine should print on the screen, but it should also be usable on a printer. That means you are not allowed to use any cursor movement keys. If the user enters 12.5, 134.56, .0026, 23, and 1.234, the computer prints out:

```
 12.5
134.56
   .0026
 23
  1.234
```

Problem #21-4: REM Remover

Jim Speers (Niles, MI) has sent a five line subprogram which may be added to the end of another program beginning at line 55000. When the user types GOTO 55000, this subprogram deletes all lines beginning with REM or a colon. Jim uses this subprogram in debugging. Any trace statements to print variables or to STOP the program may be preceded by a colon. They will be executable, but when debugging is done, they may be removed simply by typing GOTO 55000. Can you come up with such a program?

This month we have readers' solutions to the May 1985 *Commodares. Problem #17-1: Decimalizing Dates* brought a grand total of three responses. The problem is not really very difficult, and it is a "real world" problem. Business programs typically use a decimal form of a date rather than a "month, day, year" format for internal calculations. The solution from Jim Speers (Niles, MI) is listed below. Jim claims that mid-1985 should be listed as 1984.5 since he considers the first year as 0. Midway into the first year should then be 0.5. We'll leave it up to you to modify the program if you want mid-1985 to be shown as 1985.5.

```
1 REM
```

```
2 REM PROBLEM #17-1: DECIMALIZING DATES
3 REM SOLUTION BY JIM SPEERS
4 REM
100 DIMD(12),I$(3):FORI=1TO12:READD(I):N
EXT
110 DATA0,31,59,90,120,151,181,212,243,2
73,304,334
120 PRINT"[CLEAR][DOWN]ENTER DATA (E.G.
10/5/1985)";:INPUTA$:K=1:FORI=1TOLEN(A$)
130 Z$=MID$(A$,I,1):IFZ$="/"ORZ$="-"THEN
K=K+1:GOTO150
140 I$(K)=I$(K)+Z$
150 NEXT:M=VAL(I$(1)):D=VAL(I$(2)):Y=VAL
(I$(3)):LY=0
160 IFY/4=INT(Y/4)ANDY/400<>INT(Y/400)TH
ENLY=1
170 DT=Y-1+((D(M)+D-LY*(A>2)-.5)/(365+LY
)):DT=INT(DT*10000+.5)/10000
180 PRINT:PRINT"THE DECIMALIZED DATE FOR
 "A$" IS":PRINT:PRINTDT
```

Jim's program calculates the decimalized date as of twelve noon, hence the −0.5 in line 70. Each hour is 0.000114 part of a year. Line 160 tests for leap years. The DATA statement stores the number of cumulative days in months prior to the given date. David Alan Wright (New Britain, CT) and James Borden (Carlisle, PA) also sent solutions. Both Jim's and David's programs allow the user to enter dates in mm/dd/yr or mm-dd-yr format.

Several readers tackled *Problem #17-2: Printer Sentinel.* The solution from Brian J. Wilcox (Coldwater, OH) is listed. Since his program is similar in approach to many others, I concluded that it must work—on some printers at least. It doesn't work on mine. A modified version of Brian's program is also shown, which works on my Epson MX-80 printer with a serial-to-parallel converter.

```
1 REM
2 REM SOLUTION TO COMMODARE #17-2
3 REM      PRINTER SENTINEL
4 REM SUGGESTED BY BRIAN J WILCOX
5 REM
10 PRINT CHR$(147)
20 GOSUB 100
30 PRINT"THE PRINTER IS ON"
40 END
100 OPEN4,4:PRINT#4:CLOSE4:IF NOT(ST AND
 -128) THEN RETURN
110 PRINT"[HOME]PRINTER IS NOT ON LINE"
:GOTO 100
```

```
1 REM
2 REM SOLUTION TO COMMODARE #17-2
3 REM      PRINTER SENTINEL
4 REM         MODIFIED
5 REM
10 PRINT CHR$(147)
20 GOSUB 100
30 PRINT"THE PRINTER IS ON"
40 END
100 PRINT"[HOME]THE PRINTER IS NOT ON":O
PEN4,4: PRINT#4
105 PRINT#4:IF NOT(ST AND -128) THEN
 PRINT   "[HOME]                    ":R
ETURN
110 CLOSE 4:PRINT"[HOME]PRINTER IS NOT O
N LINE":GOTO 100
```

Even when my printer is off, the computer can sent it one message without the STatus variable indicating any problem. If the program sends a second message, the computer sits and waits forever at the PRINT# statement. Consequently I couldn't display a "Printer Off" message if the printer was off. Instead I had to display the "Printer Is Not On" message before writing to the printer, and if the printer was on, the message was quickly erased. If the printer is off, the message remains on the screen until the printer is turned on (or online). If neither of these routines works for your printer, you might compare ST with 0 instead of −128. Refer to the STatus description in the *Programmer's Reference Manual.*

Other solutions to this problem came from Ron Giedd (Sioux Falls, SD), Clarice A. Baker (Kamloops, BC), Jim Speers (Niles, MI), James E. Borden (Carlisle, PA), and Lonnie Welch (Independence, MO).

FREE!

BONUS GIFT WITH ANY SOFTWARE ORDER — AN EXCELLENT PUBLIC DOMAIN VERSION OF AN ARCADE HIT GAME ON ITS OWN DISK.

ARCADE

Archon	28.95
Archon II Adept	28.95
Beachead II	25.95
Bruce Lee	24.95
Castles of Dr. Creep	20.95
Championship Loderunner	24.95
Computer Quarterback	28.95
F15 Strike Eagle	23.95
Flight Simulator II	35.95
Impossible Mission	23.95
International Soccer	19.95
Kennedy Approach	23.95
Micro League Baseball	28.95
Statistics Disk (1984)	13.95
Mig Alley Ace	23.95
Night Mission Pinball	21.95
One On One	28.95
Pitstop II	27.95
Questron	27.95
Raid Over Moscow	27.95
Solo Flight	23.95
Strip Poker	20.95
Tournament Tennis	17.95
Zaxxon	24.95

ADVENTURE AND MIND GAMES

Amazon	23.95
Adventure Construction Set	37.95
Adventure Master	27.95
Below the Root	18.95
Bridge 4.0	17.95
Cutthroats	28.95
Dragonworld	23.95
Fahrenheit 451	23.95
Felony	23.95
Hitchhikers Guide	25.95
Infidel	25.95
Monty Plays Scrabble	27.95
Murder By The Dozen	23.95
Party Quiz	49.95
Party Quiz Refills	17.95
Rendevous With Rama	23.95
Robots of Dawn	27.95
Sargon III	34.95
Seven Cities of Gold	28.95
Starcross	26.95
Suspended	26.95
Swiss Family Robinson	18.95
Trivia Fever	27.95
Ultima III	39.95
Witness	25.95

EDUCATIONAL PROGRAMS

Agent USA	21.95
Cave of the Word Wizard	24.95
CBS Success w/Math Series	ea. 18.95
Evelyn Wood's Dynamic Reader	35.95
Harcourt Brace Computer SAT	54.95
Kidwriter 19.95 Memory Manor	19.95
Muppet Learning Keys	49.95
Spell It	33.95
Spellicopter	27.95
Story Machine	22.95
Word Attack	33.95

WORD PROC.-DATE BASES-SPREADSHEETS

Batteries Included Home Organizers	
Checkbook	20.95
Electronic Address Book	20.95
Home Inventory	20.95
Mail List	20.95
Photos/Slides/Movies	20.95
Recipes	20.95
Batteries Included Home Pak	34.95
Calc Result Advanced	69.95
Calc Result Easy	35.95
Complete Personal Accountant	54.95
Consultant (Delphi's Oracle)	68.95
Cut and Paste	36.95
Financial Cookbook	35.95
Fleet Systems	56.95
Magic Desk I	35.95
Mirage Data Base Manager	67.95
Mirage Word Processor	56.95
Net Worth	55.95
Paperclip w/Spellpack	82.95
PFS File	55.95
Practi Calc	34.95
Practi File	34.95
Superbase 64	52.95
Total Text 2.6	31.95
Total Mailing List & Labels	16.95

GRAPHICS

Animation Station	63.95
Doodle	28.95
Future House Light Pen	23.95
Moviemaker	37.95
Pen & Peripheral Vision	41.95
Print Shop	31.95

MISCELLANEOUS

Bulk Discs - Quantities of 10	12.95
Compuserve Starter Kit	21.95
Fast Load	27.95
Maxell Disks - Box of 10	16.95
Pro-Tech Vinyl Covers	5.35
Simon's Basic	38.95
Tac II Joystick	9.95
Vidtex	21.95
VIP Terminal	29.95

FLOPPY HOUSE

16 S. Chestnut Street, Box 308, Palmyra, PA 17078

HARDWARE AND PERIPHERALS

B.I. 80 Column Display	134.95
Cardco +G Interface	69.95
Commodore 801 Ribbons	7.95
Commodore Hardware	CALL
Indux Disk Drive	279.00
Okinate 10 w/Plug and Print	199.00
Okinate 10 Blk & Wht Ribbon	4.75
Okinate 10 Color Ribbon	5.25
Panasonic KXP1091 Printer	279.00
Prowriter 7500 Printer	229.00
Prowriter 8510 AP Printer	299.00
Prowriter 8510 Ribbons	5.95

LEROY'S CHEATSHEETS $2.95

Basic
Blanks (3)
Calc Result Adv.
Calc Result Easy
Comal (1.4)
The Consultant
Doodle
Easy Script
Flight Simulator II
For the Beginner
Hes Writer
Logo Sheet 1
Logo Sheet 2
The Manager
Multiplan
Paperclip
Practicalc
Printers
- Epson RX80
- Gemini 10X
- 1525/801/
Quick Brown Fox
Simon's Basic
Speedscript
Sprites Only
Superbase 64
Terminals
- Smart 64
- Vidtex
- VIP
Word Pro 3/Plus

SPECIALS

Cardco Numeric Keypad
32.95
Compuserve Starter Kit/Vidtex
39.95
S-Games (Formerly Hes Games)
11.95

TOLL FREE
1-800-633-8699

In Pa. Call
A.C. 717 838-8632

ORDERING & TERMS

- VISA & M.C. accepted with no charge in the continental U.S.A.
- $2.50 shipping charge on orders under $100.00
- Pa. Residents add 6% Sales Tax
- Prices subject to change w/o notice
- 48 hour shipping for cashier check - Money order - charge

Reader Service No. 143

The most compact solution to *Problem #17-3: MID$ Statement* came from Ken Karow (Chicago, IL). Ken included some sample strings for testing the routine. The value of N tells how many characters of Y$ are used to replace characters of X$ starting at position S of X$. This can be a very useful routine.

```
1 REM
2 REM SOLUTION TO COMMODARE #17-3
3 REM        MID$ STATEMENT
4 REM   SUBMITTED BY KEN KAROW
5 REM
100 Y$="ABCD":S=4:N=3
110 X$="1234567" : GOSUB200 : PRINT S,N,
X$:INPUT"START, NUMBER";S,N:GOTO 110
200 IF N=0 OR N>LEN(Y$) THEN N=LEN(Y$)
210 X$=LEFT$(LEFT$(X$,S-1)+LEFT$(Y$,N)+M
ID$(X$,S+N),LEN(X$)) : RETURN
```

The most popular *Commodare* was *Problem #17-4: Number Speller.* Solutions ranged from the reasonably short one by Rob Lackey (Albuquerque, NM) listed below to multipage programs.

```
1 REM
2 REM SOLUTION TO COMMODARE #17-4
3 REM       NUMBER SPELLER
4 REM   SUBMITTED BY ROB LACKEY
5 REM
10 DIM D$(30):FORI=1 TO 30:READ D$(I):NE
XT
20 INPUT"[CLEAR]ENTER NUMBER (<10000)";N
30 A=LEN(STR$(N))-1:ON A GOTO 90,60,50,4
0
40 N$=D$(INT(N/1000)+1)+" "+D$(30)+" "
50 V=(INT(10*(N/1000-INT(N/1000)))+1):IF
V>1THEN N$=N$+D$(V)+" "+D$(29)+" "
60 T=N-100*INT(N/100):IF T>19 THEN 80
70 N$=N$+D$(T+1):GOTO 100
80 N$=N$+D$(INT(T/10)+19)+" "
90 N$=N$+D$(N-10*INT(N/10)+1)
100 PRINT N$:END
110 DATA "",ONE,TWO,THREE,FOUR,FIVE,SIX
120 DATA SEVEN,EIGHT,NINE,TEN,ELEVEN
130 DATA TWELVE,THIRTEEN,FOURTEEN
140 DATA FIFTEEN,SIXTEEN,SEVENTEEN
150 DATA EIGHTEEN,NINETEEN,TWENTY
160 DATA THIRTY,FORTY,FIFTY,SIXTY
170 DATA SEVENTY,EIGHTY,NINETY,HUNDRED
180 DATA THOUSAND
```

Rob's program handles numbers up to 9999. Other solutions went up to billions and beyond without much additional coding. Perhaps you can modify Rob's program to handle even larger numbers. This program might be useful for writing checks. I have used this type of program with a speech synthesizer. The speech synthesizer pronounces words which are sent to it in PRINT statements, so multidigit numbers must first be converted into text strings to be spoken properly.

Solutions to these last two problems came from the following readers not mentioned earlier:

Mac Reiter (Mustang, OK)	Steve Werkman (Perrysburg, OH)
M.B. Lanphear (Cheyenne, WY)	William Manganaro (Ctr. Moriches, NY)
Thomson Fung (San Diego, CA)	William Binder (Northville, MI)
Paul DeLuca (Bradford, MA)	Jerry Anderton (Florence, AL)
Ed Taylor (Waukesha, WI)	Tony Herrington (Jeffersonville, IN)
Carlton Burton (Easton, TX)	Sheldon C. Wotring (Palmerton, PA)
R.R. Goings (Washington, IL)	Michael J. Strawn (Warren, MI)
Darin Jett (Paducah, KY)	Paul A. Dobransky (Fombell, PA)
Crile Carvey (Salinas, CA)	Richard Oberle (Columbus, OH)
Clay R. Reed (Edwards, CA)	Nancy Balfour (Honolulu, HI)

Special thanks to Ricardo Chan from the Republic of Panama and to Daniel R. Propst from Caracas, Venezuela for their letters and programs. It is exciting to see how far-reaching Commodore computers and *Ahoy!* magazine are.

One final note. Lonnie Welch (Independence, MO) gave one example of a mandatory space in BASIC. This statement using the logical OR of variables F and Q results in a Syntax Error. Do you see why?

```
10 F=0:Q=1:PRINTFORQ
```

Have you come across any other situations where BASIC requires a space? If so, let us know. This month's problems should keep you busy for a while. See you next month. ☐

AUTO-GEN

Automated Generation of DATA Statements on the C-64

BY DAVID A. JONES

Incorporating machine language routines into BASIC programs is often used to increase program execution speed. Sometimes the machine language is a separate program called and loaded by the main BASIC program. This requires two or more programs always to be on the disk, and making duplicates can become a chore. A common alternative is to store the machine language program as a series of DATA statements. This is a tedious task if these statements are typed in by hand. While the routine is under development, or should the need to modify it later arise, one simple change could necessitate retyping the entire sequence of DATA statements. Some form of automation is in order here, and the *Auto-Gen* utility program is my solution.

WHAT IT DOES

Once a machine language program is in memory, usually from the output of an assembler, just load and run *Auto-Gen* and the DATA statements will be automatically generated and written to disk for you. The program prompts you for the addresses of the first and last memory locations to be saved. These may be entered in decimal or, if preceded with a $ (dollar sign), in hexadecimal. Next you are asked for the starting BASIC line number. (Be sure not to use the same line numbers that are used by the main portion of the program under development.) You are then asked for the number of data items desired per line, and finally the name of the output file you want to create.

While the program is running, it displays the line number being generated to inform you of its progress.

HOW IT WORKS

A quick review of how a line of a BASIC program is stored in the computer will help us understand the task to be accomplished. The normal starting location for the C-64 is 2049 ($0801). The byte here plus the following one make up a link or pointer to the beginning of the next line of the program. This link is the "low byte first, high byte second" format typical of 6502 machine language. This address can be converted to decimal by multiplying the second, or high byte, by 256 and then adding the first, or low byte, to the product. Conversely, if we have the decimal value we can obtain the two bytes by dividing by 256. The integer quotient is the high byte, and the remainder is the low byte. Our formula will be HB=INT(NL/256): LB=NL−(INT(NL/256)), where HB is the high byte, LB is the low byte, and NL is the address of the start of the next line.

The next two locations hold the number of the current program line. Again, the number is stored in low byte-high byte format and the same formula is used to convert to and from decimal.

The rest of the line contains the BASIC program statements. Each BASIC keyword is stored as a token, i.e., reduced to a one byte symbol. This has a twofold purpose. One is to reduce memory requirements and the second is to simplify interpretation during program execution. For our needs, the token for "DATA" is 131 in decimal ($83 in hex). Spaces, commas, and literals (numbers or letters representing their own value as opposed to variables) are stored in their ASCII form. The numbers 0 through 9 are 48 through 57 respectively. A comma is 44 and a space is 32. The end of the line is indicated with a zero byte, and the last line of the program is terminated with three consecutive zero bytes.

With this limited knowledge we can construct our own BASIC "DATA" statements. Since one numeric data item can have a maximum value of 255, we will allocate one memory location for each digit or three locations per item. Each item must be separated by a comma, so add one per item except for the last one on a line.

Include in our tally 2 for the link to the next line, 2 for the line number, 1 for the DATA token, and 1 for the terminator, and we have the following formula. 2+2+1+4*ND+1 = the number of memory locations required to store one program line. ND is equal to the number of data items desired on each line. Add this number to the address of the beginning of the current line and we have the address of the start of the following line. NL will be the variable in our program that holds this address.

If we want to increment the line number by 10 we simply add 10 to the previous line number. LN is the variable here. When we reach the end of the program add two zero bytes to the one that terminates the last line and our work is almost done.

A program stored on disk or tape has as its first two bytes an address indicating where the program should be loaded. This is low byte-high byte format again and for our purposes the address is a constant. The C-64 uses 2049 ($0801) and the VIC 20 1025 ($0401).

That's about all we need to know to create a file of DATA statements directly on disk.

We assemble and load our machine language file into memory, note the first and last memory locations used, and we're ready to begin.

A loop that starts with the first data location and increments by the number of data items per line until the

last location is reached will be the main loop of our utility. This loop keeps track of the start locations of each line and the line numbers themselves. It writes the first part of each line to disk up to and including the token for DATA and additionally writes the line number to the screen to show the user the progress of the program.

Inside the main loop a smaller loop, incrementing by one up to the number of items per line, creates each item for the DATA statements. Internal to this loop is a third loop to step through the conversion of data items into ASCII characters and write them to disk.

Last is a counter to determine where to insert commas. We don't want a comma after the last item on a line.

Prior to entering the main loop of the program, the disk file is opened and the program load address is written. Each iteration through the loop writes a program line and at the end of the loop the program terminator is written and the file is closed.

Two subroutines are used. One does the error checking on the disk file and the second is to allow hexadecimal inputs for the first and last memory locations to be saved. Since the main function of this program will be saving machine language routines, a hexadecimal input will probably be the norm. Hexadecimal input numbers must be preceded with a dollar sign to distinguish them from decimal.

Once the file is on disk it can be edited just like any other, or appended to an existing file.

Existing programs that have an unequal number of data items per line or that for some other reason need to be reformatted can also benefit from this program. Simply load the old program and let it POKE in the data. Then load *Auto-Gen* and let it do its work. Finally reload the original program, delete the old DATA statements, and append the new. The result is a neatly formatted program with an equal number of DATA items on each line.

VARIATIONS

I like my program listings to be neatly formatted and easy to read. Consequently, I have no reservations about including spaces and blank lines to achieve this end. Lines 1560, 1580, and 1470 in the program would be the ones to change if you absolutely must have files without leading spaces on data items less than three digits.

The choice of the number of data items on each line is arbitrary, but a few points should be considered. 10 is a nice round number and helps us count items easily. 16 is a good choice when working with machine language as it allows quick conversion to hexadecimal. 17 fits the maximum number of a C-64 program line and 19 does the same for a VIC 20.

Incidentally, the sample output of this program is the first part of the DOS Wedge 5.1. It was created by saving from 52224 to 52273. To store the complete wedge use 53081 as the end address. Don't be surprised at the length of the resulting program, though. Using DATA statements to store machine language programs is most inefficient. ☐ *SEE PROGRAM LISTING ON PAGE 89*

DRAGON TYPE

For the C-64

By Bob Spirko

Computer programming offers the frustrating paradox of not lending to one's typing skills. After months or even years of programming, many of us still cannot type with any speed. This is especially apparent in telecommunications, where each party takes turns plodding letters on the screen. And when it comes to word processing, we suffer the same inertia.

Fed up with my own keyboard struggles, I went and bought a typing-drill program. This was a neat setup where you shot down a flying saucer by correctly typing in words. After each session, your speed, accuracy, and score were given. This, along with color and sound, helped to assuage the drudgery of practice. And the drills worked. In a few weeks I was typing faster and more accurately. But in time, I became bored with typing the same words. Besides, wouldn't it be better to type a variety of words? Unfortunately, I could not change the words, since the program was copy-protected.

Keeping this in mind I wrote my own typing-drill program. It includes the best features of the other, but it allows you to change the words. This provides you with an unlimited number of skill levels. If you find the words that I chose too difficult or too easy to type, or you simply become weary of them, replace them with your own. With *Dragon Type* you can also practice typing symbols: it accepts ASCII codes 32 to 90.

Nor did I stop there, I replaced the flying saucer and laser gun with a dragon. Using the magic of machine language, I brought the little beast to life so that it could run across the screen and eat words. How well he's fed, though, is up to you, for he can only eat those letters that have been correctly typed.

Dragon Type consists of two programs: the first is in machine language and the second in BASIC. Type in and SAVE both programs. Now RUN the first program. After the machine language is LOADed into memory, the second program is automatically LOADed and RUN. Then the fun begins.

First, the dragon does a quick warm-up; then you're asked to choose a level. There are three animation speeds. While a novice can use the fast level, he may feel intimidated by a frantically paced sprite. On the other hand, a good typist might find that a slow sprite cannot keep up with his speed. Once your choice is selected, words fill the screen.

Now enters the dragon. He emerges from the left side of the screen, races to the first word, and stops. Poised over the first letter, he snaps his jaws wide open and awaits your command. Tap in that letter, and the dragon gulps it down and moves on to the next letter. Continue typing and the letters disappear as the voracious creature munches them (he eats spaces, too). Press the wrong key and a beep is sounded, signaling an incorrect entry. (You can abort the meal by pressing f7.) At the end of each line, he speeds off the screen only to reappear on the next line, ready for another course. Once your dragon consumes all the words, the screen displays your speed and the number of typing errors. Also shown is your score and your highest score; these reflect both your speed and accuracy. To go another round, just press the space bar. Pressing RETURN ends the program. □

SEE PROGRAM LISTING ON PAGE 88

FILE LOCK

Scratchproofing for Your C-64 Programs

BY DON LEWIS

How many times have you scratched a file, and then immediately regretted it? If you're like me, I'll bet a lot of times. Lately there have been several UNSCRATCH utilities published, but that's a pound of cure. What's really needed is an ounce of prevention and that's where *File Lock* comes in.

File Lock provides a scratchproof feature for the 1541. Built into the DOS, but kept secret by Commodore, the 1541 has the ability to lock out the scratch command on a file by file basis. This LOCK/UNLOCK is not supported by any built-in commands in the 1541, but is available through a utility like *File Lock*.

File Lock has a very convenient user interface. When you run *File Lock*, the program prompts you to insert a disk. It then reads the directory of file names from the disk. Two large boxes appear on the screen with the file names in them. Below the boxes is the command line:

The command line has the following commands available:

SCRATCH	– scratch files
LOCK	– scratchproof files
UNLOCK	– remove lock
PAGE	– advance to next screen of files
BOOT	– restart program
QUIT	– exit to BASIC (READY)

Choose the command you want by typing the first letter of the command word, for example, L for LOCK. After you have made your choice, a small arrow appears on the screen next to the first file name. Files are selected by moving this arrow around the screen with the cursor keys, HOME key, and RETURN key. Files selected are highlighted in reverse video. Hitting the left arrow key will cancel all the files selected and return you to the beginning of the program.

After selecting the files, operation is started by hitting the first letter of the desired operation: S for scratch, L for lock, U for unlock.

File Lock works by modifying a bit in the file type byte in the directory entry for that file. (See page 67 of the 1541 manual.) As an example, a *PRG* file has a file type of \$82(1000 0010) if it is not locked. When locked, bit 6 is set, yielding \$C2(1100 0010).

Locked files are marked in the directory listing by a < symbol next to the file name. Once a file has been locked, the 1541 will send the error message 'FILE NOT FOUND' if an attempt is made to scratch the file.

File Lock tends to be a bit sluggish in its operation since it is a BASIC program. However, when it is compiled with a BASIC compiler such as *Blitz!*, its performance is greatly improved. If you like *File Lock*, I would recommend that you *Blitz!* it for the best performance.

As with my other program in this issue (*Fastnew*, p. 33), I wish to acknowledge the assistance of my colleagues at R. J. Brachman Associates, Inc. (Havertown, PA) in developing this program. Please address any correspondence to D. Lewis, P.O. Box 521, Folsom, PA 19033. Include a SASE if you wish a reply. ☐

SEE PROGRAM LISTING ON PAGE 93

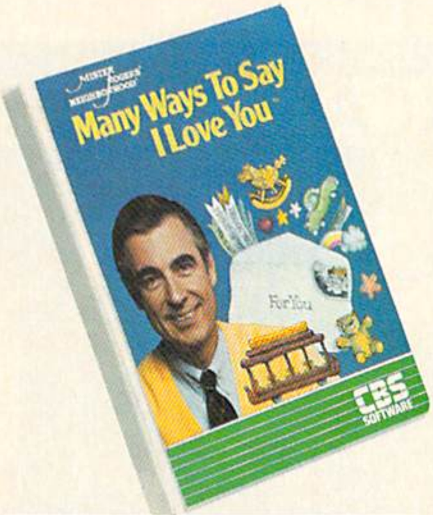
MANY WAYS TO SAY I LOVE YOU
CBS Software
Commodore 64
Disk; $29.95

Fred Rogers has used music and conversation to teach children to value themselves and their families for almost 18 years on *Mister Rogers' Neighborhood,* the longest running program on public television. In an effort to expand the relationship between parents and children, *Many Ways to Say I Love You* gives children an electronic forum to creatively display their affections.

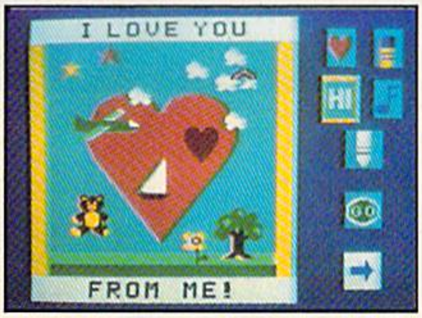
All the songs in the program are easily recognized, either popular folk tunes or regulars from the show. My two year old had no problem singing along with *Won't You Be My Neighbor* and *It's Such a Good Feeling.* Before long she had picked up *Twinkle, Twinkle, Little Star* and *Frère Jacques.*

Ah! But what are the songs for, you ask? *Many Ways to Say I Love You* helps a child design and deliver an electronic greeting card, complete with moving pictures and music. The child can pick a message like "I think you're special" to appear at the top of the card type in text on the bottom. In this way, each card can be personalized.

The graphics are great. Choice of "stickers" includes animals, trees, flowers, moon, stars, houses, a castle, and residents of *Mister Rogers' Neighborhood* (including the trolley, Mr. McFeeley, and several characters from the land of make-believe). All the stickers will perform some action while the card is being "delivered." Before choosing pictures to illustrate the card, a background and a border design can be chosen.



Lets tykes design and personalize (but not print) electronic greeting cards.



Child picks background, border, more.
READER SERVICE NO. 102

There are two levels of difficulty: a full frills card or a simplified one with just background, stickers, and music. The simplified version is good for youngsters who have not yet learned to read.

The program does have an option to save cards to disk. Up to 24 cards can be saved on your master disk; however, I wouldn't take the chance. CBS takes pains to explain the buyer's rights in the program, and these rights include making a backup.

While I'm skeptical as to whether this well-designed program will open any heartfelt chats between parents or children, it does offer a chance for closer interaction. Any parent who buys it should be prepared to spend time listening to cards. There is no printout option, so you won't be able to send greetings to friends and family around the country.

If nothing else, my daughter has learned to sing two new songs. And the youngster I borrowed from downstairs showed her mom that computers can be used for something besides shooting aliens and learning to add and subtract.

CBS Software, One Fawcett Place, Greenwich, CT 06836 (phone: 203-622-2500). *—Cheryl Peterson*

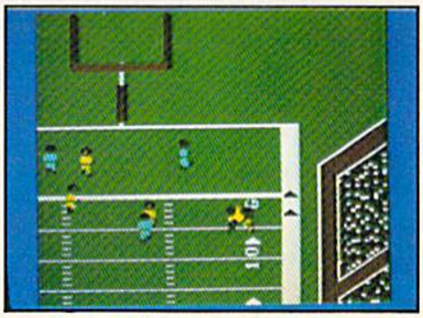
ON-FIELD FOOTBALL
Gamestar
Commodore 64
Disk; $29.95

The popularity of football has leveled off on television, but there's a real boom underway in home computing. *On-Field Football* is only one of three pigskin programs scheduled for release in time for a fall kickoff, and at least one more is due early in 1986.

Unlike the other newcomers, which stress authenticity, *On-Field* emphasizes freewheeling fun. The media hype of the Super Bowl is light-years away from this brand of football.

The inspiration for designers John Fitzpatrick and Scott Orr is not the National Football League, but rather the sandlot and pickup football games which fill youngsters' fall and winter afternoons. This is four-against-four, not big-time football.

Don't expect to be able to mimic all the latest wrinkles in pro and college football. The coaches in this one- or two-player contest can choose from a variety of offensive and defensive moves, but the true simulations like *Super Bowl Sunday* (Avalon Hill Game Company) and *The World's Greatest Football Game*



Players have individualized strengths.
READER SERVICE NO. 103



MASTERTR
ARCADE-STRATEGY-ACTION
Sure Chart Winners
VEGAS JACKPOT
One of the most faithful computer simulations ever produced of a fruit machine. It incorporates the hold, nudge, gamble and shuffle features found in the real thing.
SPACE WALK
A satellite retrieval game that includes 16 levels of "low pressure" death awaiting the unskilled astronaut.
BMX RACERS
A sure chart winner. You will need skill and fast reflexes to beat this one! You can ride on 5 different courses, each course ranges from 8 to 25 different scrolling screens.
CHILLER
Opened to Five Star Reviews in Europe recently this is surely one of the most challenging and unusual games available for the C64. Written by "Commodore Programmers of the Year" Richard and David Darling. The game has five different spectacular screens with hidden twists — if you get this far you have to retrace your steps through another five of increasing difficulty.
MIND CONTROL
Another unusual arcade — strategy game in which the players strive by weird and wonderful techniques to achieve a state of miniaturisation and hence enter Zyco's brain and the game continues in another dimension.
MAGIC CARPET
An imaginative three screen multi level arcade adventure that uses very high resolution graphics and will test the abilities of the most experienced games players.
1985 THE DAY AFTER
A scrolling multi screen game that takes Commodore graphics to a 'new' state of the art level previously seen only in a handful of other games. Playability is a keyword on this 'five star game'!
KIKSTART
A one or two player game with split screen scrolling graphics that realistically simulates off road riding on eight different and challenging courses.
More Adventures of BIG-MAC the Mad Maintenance Man
Multi screen arcade action with real twists needing a steady hand and a bold strategic game play. It will keep your interest alive for hours.
challenger
Neat variation on a city bomber theme using excellent scrolling graphic routines that is extremely addictive.
MAGRA and GAMES CREATOR $19.99
$9.99
SPECTACULAR
HIGH
QUALITY SOFTWARE

READER SERVICE INDEX

The publisher cannot assume responsibility for errors in the above listing.

(Epyx) boast significantly heftier playbooks.

Joystick control makes *On-Field Football* exceptionally easy to learn and play. The coaches create each play by following a series of onscreen prompts. On offense, the gamer pushes the joystick forward, back, left, or right to select one of five possible formations: "I" right (run), "I" left, split left (pass), slot right (pass), and kick. Then the offensive coach individually programs the video athletes who are going to run with the ball or catch it. Finally, the computerist can revise the play at the line of scrimmage by calling an audible. Pressing the action button when prompted to do so changes the instructions for the halfback or the wide receiver, depending on whether it is, respectively, a running or passing play.

Coaches issue orders to the defense in the same way. The display prompts the gamer to choose separate options for the rush and secondary. The defense can call an audible to change the pass coverage at the last second.

Once both teams complete play-selection, the action button snaps the ball to start the action phase. The offensive coach controls the passer with the joystick and employs a simple system to hand off the ball on running plays or choose between the two eligible receivers when putting the pigskin into the air. The defensive coach guides the free safety with the stick and can even try to blitz the quarterback for a big loss.

The players are more than just animated cursors in *On-Field Football.* As in the same publisher's *On-Court Tennis,* the coaches pick individual players from a roster of possibilities prior to the opening whistle. For instance, there's a choice between the slick-passing "Pinpoint" Pepper and the more mobile "Scrambles" Smith. The teams can bring in replacements for tired players between the third and fourth quarters. It's too bad that the same concept wasn't extended to include members of the defense, who remain anonymous. This makes the game much less involving when the other team has the ball.

The playfield is oriented so that the team with the ball always moves up the screen against defenders protecting the goal line at the top. When possession changes, the teams swap places.

Gamers traditionally favor football programs in which the action moves side-to-side, but this attempt is more successful than most previous tries. The traditional complaint against vertical gridirons is that they don't provide enough room for passers, runners, and receivers to maneuver. The designers of *On-Field Football* have made the players as small (relative to the size of the field) as practical to minimize this. Most computerists will still prefer horizontal football fields, but the overall effect isn't bad.

The most useful of a batch of special features lets players choose between regular 15-minute quarters and shorter 5-minute ones. The program also allows each coach to call three time-outs during the game. It is also possible to "pause" play entirely when the phone rings or nature calls.

Like other Gamestar titles, *On-Field Football* takes practice to master. (Not, however, quite as much as *Starbowl Football,* also by Gamestar.) Computer sports fans who want an action-oriented football contest, rather than one which relies on pure strategy, are the most likely to enjoy this clever game.

Gamestar, Inc., 1302 State Street, Santa Barbara, CA 93101 (phone: 805-963-3487). *–Arnie Katz*

COMPUTEREYES

Digital Vision Inc.
Commodore 64
Hardware, disk; $129.95

There is an old saying that a word is worth a millipicture. Nothing brings this truth home more clearly



than working with digitized images. On the Commodore 64 a word is actually worth even less, at most an

eighth of a millipicture. If color is considered, the fraction can be as small as one tenth. Of course, the words we are talking about are eight bits or one byte each. Nevertheless, the veracity of the adage is brought home with a vengeance.

The Computereyes Video Acquisition system adds vision, without foresight, to the ever-growing list of possible applications for the C-64. The system will allow you to capture those fleeting images and translate them into the 8-10,000 bytes needed to describe a bit mapped image.

The physical appearance of the package belies its impressive performance. Computereyes is an excellent example of what a clever design with good programming can accomplish when coupled with the capabilities of the 64.

In keeping with this introduction we have included several screen shots which illustrate the capabilities of the system. The Chinese plates show how Wayne Schmidt used the information contained in a digitized *DOODLE!* image. The color and fine detailing were all added afterwards. We should all be familiar with the original on which the Washington head on page 68 is based. The young lady below it is the cover model from another of this company's publications.

THE HARDWARE

The Computereyes module is a small black box that plugs into the user port. It serves as an interface between any standard video signal and the computer. Suitable sources are video cameras, VCR's, video disc players, or TV's with video outputs. The video signal is fed to a standard



Color and fine detail were added (left) to a digitized DOODLE! *image.*
READER SERVICE NO. 105

RCA type jack on the back of the module. One of the two controls allows for the fine tuning or synchronization to the video source. The adjustment is non-critical and easily accomplished with the supporting software. You will most likely find that the setting will not change for a particular video source.

The other control compensates for the relative strength or brightness of the signal. The brightness control has a direct effect on the appearance of the acquired image. We found the setting of this control to be critical but manageable. Small adjustments of the brightness control have a significant impact on the final image.

THE VIDEO SIGNAL

The images displayed on a TV are redrawn 30 times a second. Each individual image consists of 512 horizontal lines. These are sequentially drawn from top to bottom of the picture tube. Creating an image in this manner is called scanning. Each image or screen is built up from two passes of 256 lines each. The first pass fills the entire screen; the second fills in the gaps between the lines of the first scan. This arrangement is called interleaving. It follows that the ideal TV picture would be able to resolve 512 discrete verticle points. However, in practice this is not so.

The capabilities of most TV's are about half of the theoretical maximum. The newer "monitor" receivers are somewhat better, on the order of 350 lines. The Commodore 1701 and 1702 monitors fall into this class. Home VCR's are able to resolve about 260 lines at this time, although better equipment is in the works. Laserdisc players generally exceed the resolving capabilities of the best TV sets.

SELECTING A VIDEO SOURCE

Since the Computereyes module merely acts as an interface, you will have to provide it with a standard video signal. If you already have a home video system, with a color camera, then you're all set. In comparison to the overall cost of a complete video system, the $130 price of this package is almost too good to pass up. If you do not have a home video system, a black and white video camera is your best bet. These are available as no frills units for use with closed circuit television security systems. The Ikegami model ITC-40 Surveillance camera is a high quality unit of this type. It is available as an option from Digital Vision. The package price of camera and module is $350. Bear in mind that surveillance cameras are truly no frills devices, lacking any type of viewfinder or sound pickup.

A home VCR is a less than ideal video source for this application. The process of acquiring a detailed image takes quite a few seconds and several tries as the brightness is fine tuned. A stationary image must be provided for these several minutes. The freeze frame mode of a VCR provides less resolution than normal operation. Also, the VCR will automatically start running after five minutes or so to prevent the wearing of a hole through the tape. A laserdisc player should make a very good source. These devices provide a very high quality single frame image for unlimited intervals.

A color video camera will work well. You may want to use its associated VCR as an intermediary device to avoid the cost of an additional cable and adapter. We did all of our tests with the Ikegami camera.

A BIT OF A PICTURE

The Commodore 64 has the ability to create a graphic image composed of 320 horizontal dots by 200 vertical dots. This is a bit-mapped image, which means that each one of these 64,000 dots or pixels corresponds to a single bit of computer memory. This is where we get the 8,000 bytes or words mentioned above. A total of 16 colors may also be displayed. The color information requires another 1,000 bytes of memory.

An alternate display mode permits the display of only 160 horizontal dots with the same number of vertical dots. The tradeoff for this multicolor mode is increased flexibility in the display of color. The memory requirements of the bit map for this mode are the same as above. The increased color capabilities call for 2,000 bytes of

storage. (For additional discussion of Commodore 64 bit mapped graphics, see the October and November 1984 issues of *Ahoy!*)

As we can see, a C-64 bit map would seem to contain as much information as a TV picture. The reason it cannot create a televisionlike image is that the 64 is limited to a total of 16 possible shades or colors. A television image is capable of thousands of brightness and color combinations. Nevertheless, impressive results can be obtained, some examples of which can be found in our *Art Gallery* (see pages 34 and 35).

A number of software packages allow the user to draw or create graphic images on the 64, including *Flexidraw* from Inkwell Systems, *DOODLE!* from City Software, and *Koala Painter* from Koala Technologies. Detailed reviews of these and a number of other packages were presented with the discussions on bit mapped graphics previously mentioned. These packages are supported by the Computereyes software.



George digitized with Koala *support software, in low contrast format. With multicolor packages like* Koala, *compatible image modes are limited to low and high contrast capture, with up to four gray levels in each character cell.*

GETTING THE PICTURE

The Computereyes software allows you to capture a video image as a Commodore bit map. Three different bit map formats are directly supported. The optional software for use with the above graphics packages adds seven more formats. We will go into these in some detail.

The first step is to set the synchronization control. A built-in routine simplifies the process to just a few seconds. Once set, the sync control is left alone. The second step is to set the brightness control. A special display mode performs a continuously visible image scan every six seconds. We found that by setting the brightness control so that only the brightest highlights showed resulted in the best starting setting for subsequent operations. The operation of the package was very sensitive to the setting of the brightness control. Although the adjustment was manageable, some sort of fine tuning would have been welcome. The controls also lacked pointers or markings, a minor inconvenience. It would be a good idea to mark the position of the controls after the initial setup. This would eliminate the need for these preliminary adjustments for each subsequent use.

The package is menu-driven, with built-in help screens. It is a complete image capture utility which allows for the LOADing, SAVEing, and viewing of the captured images. The disk directory may be viewed without exiting the program; however, disk commands, such as for formatting a disk, cannot be issued. The programs are written in BASIC with the digitizer parts in machine language. Since no copy protection is used, it should be fairly easy to add whatever frills you may want. The starter package makes full use of the high resolution bit map.



Model from magazine cover digitized with DOODLE! *high contrast capture, resulting in a bit map image with three intensity levels—white, gray, and black—with maximum resolution of 320 X 200 pixels and two levels possible in each character cell.*

The quickest way to digitize an image is to use "Normal Capture." This produces a high contrast black and white image. The original picture is scanned once with all parts above a certain brightness level assigned to white. Maximum resolution is obtained at the expense of any tonal gradation. This mode is relatively fast, completing a scan in less than seven seconds.

Somewhat slower but generally more pleasing is the "4-Level Capture." The original image is scanned four times at different exposure levels. Each level is assigned to a shade of gray by turning on different numbers of pixels. The tradeoff is loss of resolution, as four pixels are required for each part of the image. A 4-Level scan requires about 26 seconds.

The "8-Level Capture" is similar to the above. Eight synthesized grey levels are formed with a corresponding reduction in image resolution. Capture time is a bit over 50 seconds.

OPTIONAL SOFTWARE

The basic software generates a pure

8K bit map without any color information. If you have the *Flexidraw* graphics package, any of the above formats can be loaded in and worked on. Good results will also be obtained with the *Flexidraw* print utility, since these are pure bit map images. For use with other packages, Digital Vision offers Compatibility System Software in support of *Koala*, *DOODLE!*, *Flexidraw*, and *The Print Shop*. Each disk tailors the captured image to the specific package.

For the hi-res packages, namely *Flexidraw* and *DOODLE!*, the above-mentioned capture modes are supported. Two additional modes are also provided. The "Low Contrast Capture" makes use of the C-64 color capabilities to generate a true five level gray scale composed of the three grays, black, and white. In this mode the full bit map resolution is preserved while still allowing for image shading. Since the high resolution bit map is used, only two gray levels will appear in any 8 x 8 pixel character cell. The "High Contrast Capture" is similar to the above, except every other gray level is used.

Since *Koala* is a multicolor package, the compatible image modes are limited to the Low Contrast and High Contrast captures. The difference is that finer gray scale resolution is obtained, since multicolor mode allows up to four colors in each character cell. As a result the *Koala* images are apt to be the most pleasing for many subjects, even with the reduced resolution of the multicolor bit map.

The latest addition is the *Print Shop* support package (see the review of the *Print Shop* in the July '85 issue). This will create images for both the *Kaleidoscope* and the *Graphics Editor* utilities. The support for the latter is very well done. A large portion of a full image is marked by a movable mask. Simply use the cursor keys to select the portion you want. The software then blocks the image into the reduced detail for the graphics editor. When selecting a source image for conversion, keep in mind the 88 by 52 dot resolution of the Graphics Editor (44 by 45 dots for 1525 printers). Images with large solid areas will work the best.

The *Print Shop* support package can be used to convert any 8K hi-res bit map image. The ones generated by *Flexidraw*, without color, are perfectly suitable. Image files created by the *DOODLE!* package will have to be stripped of their leading kilobyte of color data. To do this you simply LOAD "DDPICTURE",8,1; where "DDPICTURE" is the file you want to convert. Next POKE 45,0:POKE 46,96. Finally SAVE "PICTURE",8; where "PICTURE" is the name of the file you want to create. The resulting 33 block disk file will be a pure bit map, without the color information, ready for processing by the Computereyes *Print Shop* support package.

PRINTING THE PICTURE

The built-in print utilities of both *DOODLE!* and *Flexidraw* will do a fine job with the first three image formats. Strange results are likely if you try to print either the Low or High Contrast images. These images depend on the color information to produce the different shades of gray. The *Koala* screen dump utility will do a good job with either of the *Koala* formats. The Graphic Printer package by Michael Keryan takes into account the color information of a high-resolution bit mapped image. It is designed to work with the high density print modes of the Star Micronics Gemini series of printers or the Epson graphics-capable printers. Contact the Computerist, Inc., P.O. Box 6502, Chelmsford, MA 01824 (phone: 617-256-3649) for information about this package.

For a real personal touch, try transferring a digitized image to a T-shirt by using the Underware Ribbon. This ribbon for the Epson, Gemini, and Okidata printers contains thermal transfer ink for iron-on images. Just remember to do a left to right flip with the *DOODLE!* software before printing. The ribbons are available for $19.95 from Diversions Inc., 1550 Winding Way, Belmont, CA 94002 (phone: 415-591-0660) (see the May '85 *Scuttlebutt*).

SOME OPERATING TIPS

The techniques used for black and white photography can be applied to imaging with the Computereyes system. As such, good photography practices apply. Pay careful attention to lighting and background. If you are using a black and white camera, photographic colored filters for contrast enhancement can be quite useful.

Want to get close up? A low power magnifying lens placed directly in front of the camera lens will allow you to focus much closer than otherwise possible. The cost is minimal and image quality will be virtually unaffected.

Being able to directly view the video image greatly simplifies the setup procedure. If you are using a monitor with your computer, a simple RCA type "y" adapter and a two position video switch are all you need. Just intercept the camera signal with the "y" adapter and send it to both the Computereyes module and the video switch. The computer video output goes to the other side of the switch to complete the hookup. The Commodore 1701 and 1702 monitors are ideally suited for this purpose. Simply send the camera output to the video jack on the front of the monitor. The computer should go to the back connectors. The switch on the back panel will let you select between the two sources.

The camera video signal cannot be fed directly to the antenna terminals of a TV set. It will have to be converted to the radio frequency broadcast format. An RF Modulator, available from Radio Shack as catalog number 15-1273 for $24.95, will do the job. If you can do some of your own tinkering, then the basic RF modulator, sans power supply, is available for $9.95 as catalog number 277-221. Simply feed both the computer and the modulated video signals to the set's antenna terminals via a game selector switch. Or set one to channel 3 and the other to channel 4 and use the set's tuner.

CONCLUSIONS

Computereyes is an easy to use image digitizing system for the 64. The resulting images are remarkably detailed for such a low cost system. With care, directly usable images can be readily generated. The optional supporting software allows the images to be saved for subsequent editing by some of the more popular graphics packages. The resulting images provide a wealth of detail for manipulation in this environment.

Digital Vision Inc., 14 Oak Street–Suite 2, Needham, MA 02192 (phone: 617-444-9040).

AUTHOR'S NOTE

While working with Computereyes for this review I rapidly accumulated several disks full of images on a variety of subjects. In fact the process became rather addictive. If you would like to have these for your own use I would be glad to provide a diskful for $12 to cover copying and distribution. Just specify which format you would like. An SASE will get you a listing of available disks and their contents. Send payment to Morton Kevelson, P.O. Box 260, Homecrest Station, Brooklyn, NY 11229.

BLAZING PADDLES

Baudville
Commodore 64
Disk; $34.95

Billed as a graphics package that will work with most of the input devices available for the Commodore, I found that *Blazing Saddles* worked well with both a joystick and the Koala Pad. Although it worked with a Tech Sketch Light Pen, it was extremely erratic. While supposed to work with track balls, paddles, light pens, joysticks, or graphics tablets, it can only be used with one device each time it is booted up. If you wish to change devices during a session, you must reboot the program–a three-minute process. Quick loaders like *FastLoad* and *Mach 5* have no effect on *Blazing Paddles*.

One annoying deficiency can be found as soon as you decide to boot the program. The documentation warns not to plug or unplug devices while the program is running, but doesn't bother to tell you which input devices use which ports. Despite the warning, when the joystick failed to move the cursor I switched it to the other port. Fortunately, nothing blew up. From my experience the Koala Pad and light pen used port 1, the joystick used port 2.

The program is designed well, with most of the features we've come to know and love: sketch, fill, zoom, and draw dots, lines, frames and boxes, ovals and circles. Especially nice is the undo command. Hitting a single key cancels your last update. A spray paint mode lets you brush in color a little at a time. Subsequent strokes darken the shading.

The cut and paste window can be used to duplicate things on the screen without having to redraw them. A window's contents can be saved and reloaded using a separate filename, as well.

The screen can be cleared to any of the 16 colors. Since this is a two-step process requiring some precision, it is very difficult to accidentally wipe out your work. In fact, about the only way to really mess up is to change the background color to the same color you're using to draw

with and then return to the main menu. This is entirely possible, since you can change the background color at any point during your work by using f1.

A text mode allows you to put messages or labels in your drawings. The characters look strange, both on the screen and in the printouts. One of the companion disks, *Shapes and Fonts*, has a number of other type styles that you can use for writing messages and labels.

The $20 companion disks (*Shapes and Fonts, Science, Kid Stuff*) are probably the best recommendation for this program. These have predrawn figures that cover a variety of subjects. I used the electronic symbols from the science disk to create a circuit drawing. The resolution of printouts leaves something to be desired: high resolution. The print quality is about the same as the *Koala Painter* in low resolution mode.

The *Blazing Paddles* disk includes a few stick figures to show you how to take images out of the picture libraries and put them into your drawings. This is one area where this program shines. The sprite sized figures can be manipulated before you put them into your picture. You can flip them over, rotate them around 90 degrees, or change their color. The extra disks have some useful shapes on them (especially if you're a kid trying to impress your science teacher).

One other disadvantage that I found was not being able to exchange colors. Although switching the background colors is easy, if you want to change the color of lines you've already drawn it's a difficult task. I've only seen one graphics program with this feature, so I can't be too hard on *Blazing Paddles* for not having it. I originally chose a dark background with light lines for my circuit diagram because it was much easier on the eyes. When I got it finished, though, I realized it would look much better reversed. It took another two hours to redo the drawing. If I could have changed the background using f1 and then changed the lines to another color, it would have taken only a few seconds. Using fill on thin lines is al-

most impossible, especialy if they're diagonals.

Except for the one flaw mentioned at the beginning, the documentation is very good. Besides the manual, there is an onscreen help menu. Although it's a nice idea, by the time the help file loads you could have looked it up in the manual. Keyed to the input device you are using, it gives brief instructions on how to enter commands.

For instance, the joystick will draw on the canvas as long as the fire button is held down. To go back to the main menu, you must press f5 to take the cursor off the screen and then hit the fire button. For some reason, when using the joystick the cursor won't go over into the border.

The joystick is very slow to respond, by the way. The light pen button wouldn't work, so I had to use the control key instead. Also, the calibration on the light pen was off by about half an inch. In addition, whenever I tried to change the brush color, the background color of my drawing would change, too. Although the f1 key could be used to change it back, it was a pain in the neck. Of all the input devices used, I enjoyed the Koala Pad most. It responded quickly, and didn't give me any trouble.

This is a nice graphics package, although the input device you use will ultimately determine how satisfied you are with it. I would rate it about equal to other packages in its price range. Unless you are specifically looking for a package to use with several input devices, you would probably do just as well to buy any of the devices that comes with its own software and leave it at that. If you want to spend the extra money for the shapes disks, then *Blazing Paddles* becomes a good value.

Baudville, 1001 Medical Park Drive S.E., Grand Rapids, MI 49506 (phone: 800-824-8873).

—Cheryl Peterson

GO-LISTER

A Programming Aid for the C-64

By John K. Lunde

Go-Lister is a machine language utility that produces a printout showing which BASIC program lines are called by GOTO, GOSUB, THEN, and ON statements in other program lines.

Anyone who has ever tried to modify a program written by someone else, or who has simply lost track of where the GOTOs get to in his own program, knows the frustration of changing a line in some apparently innocuous way only to have the program begin to act strangely because the line is accessed by a transfer statement somewhere else.

This utility provides a warning when a particular line is used by several other lines in common, and shows when it is safe to 'crunch' one or more lines together.

It is also useful in analyzing programs, since knowing which lines are called by others gives you a good idea of the program's structure.

Go-Lister must be typed in using our *Flankspeed* machine language entry program. Follow the instructions on page 86 for typing and saving *Flankspeed*. Then run *Flankspeed*, and type in and save *Go-Lister*.

To use *Go-Lister*, type LOAD "GO-LISTER",8,1 and load your target program, then type SYS 49152 to run *Go-Lister*. The only restrictions are that the target program must start at the usual start-of-BASIC location (2049) and that the printer must be device #4. □

SEE PROGRAM LISTING ON PAGE 109

MOXEY'S PORCH

For the C-64

By Bob Blackmer

Moxey's Porch is an arcade-style game played on a three dimensional multistep porch.

After entering and running the game you will be asked if you want instructions. Then enter the level (1-9) that you wish to start at (level 9 is easiest).

You control Enzo as he jumps on his pogo stick. His neighbor, Moxey, a likeable fellow, has just put the finishing touches on his newly painted porch.

Using the joystick in port #2, maneuver Enzo around the porch marking up the steps. Don't let Moxey catch you as he runs around his porch trying to stop you, or the game is over.

After you mark up all the steps, you reach the next level where Moxey gets a little faster. See how long you can keep going once you reach level 1.

Moxey's Porch is a good example of the game screens made possible by using multicolor custom character sets and multicolor sprites. □

SEE PROGRAM LISTING ON PAGE 91

TOURING THE COMMODORE 128 KEYBOARD

Continued from page 54

TABLE OF CONTROL KEY FUNCTIONS

CONTROL B	CHR$(2)	Turn underline on in 80 column display
CONTROL G	CHR$(7)	Produce bell tone
CONTROL H	CHR$(8)	No effect
CONTROL I	CHR$(9)	Move cursor to next tab stop
CONTROL J	CHR$(10)	Send a linefeed without a carriage return
CONTROL K	CHR$(11)	Enable character set change
CONTROL L	CHR$(12)	Disable character set change
CONTROL O	CHR$(15)	Turn flash on in 80 column display
CONTROL X	CHR$(24)	Tab set or clear
CONTROL [	CHR$(27)	Send an ESCape character
	CHR$(130)	Cancel 80 column underline
	CHR$(143)	Cancel 80 column flash

THE REST OF THE KEYS

The top row of the keyboard has a total of 16 keys in groups of four. The four function keys on the right have been described above. The leftmost grouping includes the ESCape and the TAB keys as well as an ALT key and a CAPS LOCK key. The ALT key is intended for use with alternate character sets under program control. This will be a topic for future discussion. The CAPS LOCK key acts as a SHIFT key only on the letter keys. As with the SHIFT LOCK key, the CAPS LOCK key includes a mechanical latch to hold it down.

The next group starts with a HELP key. If this key is pressed immediately after a BASIC program error, the offending BASIC line will be displayed. Starting with the point in the line at which the error occurred, the remainder of the line will be displayed in reverse on the 40 column display and underlined on the 80 column display.

The LINE FEED key moves the cursor down without a carriage return. This is the same as the CONTROL J (CHR$(10)) described above. The 40/80 DISPLAY key is another mechanically latched button. It determines the display mode on power up or when the reset button is depressed. Latching it down defaults to the 80 column display. It has no effect once the computer is up and running. The NO SCROLL key acts as a pause button. It comes in handy for freezing a program or directory listing in its tracks. Hitting a key a second time restarts the listing. In a similar fashion, it also pauses program execution.

The next group consists of four independent cursor keys. These duplicate the operation of the cursor keys at the bottom of the keyboard. The notable exception is that it is not necessary to SHIFT for a cursor up or a cursor right. I found that I have grown so accustomed to the old cursor keys that I almost never used the new ones. Perhaps the need to reach to the top of the keyboard discourages their use.

Finally, a 14-key numeric keypad duplicates the function of the number, plus, minus, period/decimal point, and RETURN/ENTER keys. The numeric keypad and the four independent cursor keys are electrically different from the number keys and the cursor keys on the Commodore 64 subset of the keyboard. Although they return the same CHR$ code with a GET statement, it is possible to distinguish them by PEEKing an appropriate location. (See sidebar on page 52 for details.)

This hardware separation of the keypad from the rest of the keyboard is a minor disappointment to C-64 upgraders. If the keypad had been kept electrically identical to the number keys on the keyboard it would have been usable in C-64 mode as well. This would have allowed the use of all existing Commodore 64 programs with the numeric keypad. Keeping the keypad compatible could have been easily accomplished by simply paralleling the existing contacts. Actually, this would have been cheaper to implement than the scheme used. Apparently, Commodore felt the extra flexibility offered by the electrical separation was worth the extra effort.

LAST BUT NOT LEAST

Tucked away on the right side of the C-128, next to the on/off switch, is a small square push button. It is mounted flush with the surface, making accidental activation unlikely. This is a true hardware reset button. Pressed on its own, it will bring the C-128 back to its initial power up state in the current mode. In conjunction with the Commodore logo key, it will force a reset to C-64 mode. If the RUN/STOP key is held down, the C-128 will come up in the machine language monitor with the current BASIC program in memory intact! ☐

Rockets, Boats, and Pigs in Pokes

Continued from page 28

	into the flag-bit register at HR.
VZ	The lowest vertical position allowable on the screen, minus 1.
HZ	The rightmost horizontal position allowable on the screen, minus 1.
VP	The current vertical position (a number from 0 to VZ).
HP	The current horizontal position (a number from 0 to HZ).

In addition, *Pig in a Poke* uses several game-control variables:

WS$	This string contains all the fence units; segments are randomly selected to make each new fence moving up the screen.
WL*(n)*	This array determines the starting points for each difficulty level in selecting a segment of the fence to display.
RC	Row count–how many blank lines have been inserted between fences.
RT	The total number of fences added to the screen so far.
DT	The delay variable–how many seconds should pass before the fences are scrolled upward.
DL	The difficulty level, or gap-width variable–how far into the string WS$ the next fence should come from. Since gaps are narrower and fences wider the farther you go into WS$, the higher this number, the more difficult to get through the fence.
DS	The spacing between fences. It begins with five blank lines between fences, then progresses to only one blank line before the program ends.
EG	End-of-game flag. Several ending conditions set this flag, which causes the program to jump to the ending routines.
K, K*(n)*, KP*(n)*	In the main loop, these variables are used to interpret keypress data.
KP	The current direction of movement. This is used with LT*(n)* and ST*(n)* to determine whether to display the shape of a left-facing pig or the shape of a right-facing pig.

At the beginning of *Pig in a Poke*, line 5 moves the top of BASIC memory below the video block at 32768. Then lines 10 through 13 send the program to four set-up routines.

Video Memory Set-Up. Lines 600-604 tell the VIC-2 to find the video block at 32768, and where within that block the VIC-2 and BASIC can find screen memory.

Lines 610-634 assign the addresses of various registers to the variables listed above. Lines 636-637 set up the set-bit and clear-bit arrays.

Lines 640-654 set the initial values of most registers. If you want to experiment, change these at will and see how they affect the game.

Lines 690-696 disable the SHIFT/COMMODORE and RUN-STOP/RESTORE interrupts. This is because SHIFT and COMMODORE are used for left and right movement in the game, and we don't want the character set shifting back and forth; and because video memory has been relocated, a RUN-STOP/RESTORE would cause the machine to become virtually unusable.

Background Preparation. Lines 700 to 730 set up the wall string WS$ and the variable WL*(n)*, which controls the starting points for random wall segments at different difficulty levels. Line 740 sets the initial difficulty levels. DL should start at 0. DT is the number of seconds the program waits before inserting rows. A value of 0 is too fast to play. DS is the number of spaces between rows at the beginning. You have to make sure the game ends before the program drops the value of DS below 0 –this will cause a program-crashing error.

Sprite Shape. Lines 800-808 read the strings of sprite data, and interpret them so that they can be POKEd into sprite shape blocks in the correct order. If you change the shapes, make sure that each string has 24 characters and there are as many rows as are called for by the variable TS *plus 1.* If TS=9, then there must be ten strings of sprite information per shape. There must also be as many shapes as are called for by NS *plus 1*, so that if NS=1, there must be 2 whole sprite shapes.

Sprite Positioning. Line 900 sets the number of vertical and horizontal positions that will be allowed, and DIMensions the applicable arrays.

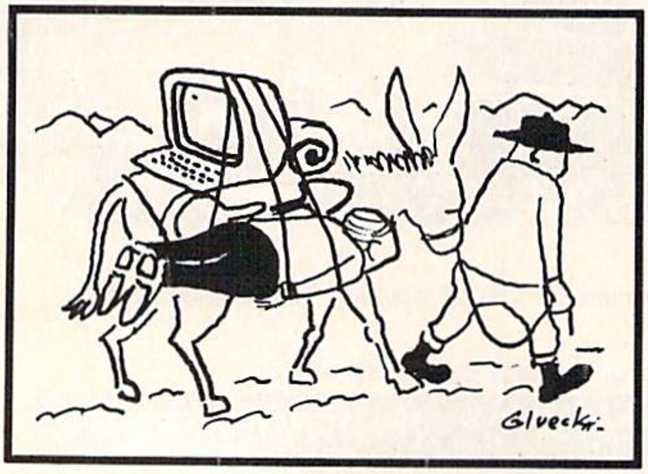
Lines 902-904 set the horizontal position values, starting with the leftmost position, 24, and increasing each position by 2. Line 906 sets the vertical positions, starting at 53 and increasing each position by 8.

Lines 920-922 set up the variables used in interpreting keypresses. KP*(n)* is indexed by the raw keypress at location 653. Values that mean nothing are set to 2; a keypress calling for left movement is set to 0, for right movements, to 1. The keypress that calls for the end of the game (SHIFT/COMMODORE/CTRL) is set to 3. These will be interpreted in the main loop.

K*(n)* has only two possible values: K(0)=–1 and K(1) =1. When indexed by KP*(n)*, these values will be added to or subtracted from the current horizontal position (HP) of the sprite.

Lines 930-931 set the initial sprite position and POKE it into the proper registers. Line 940 POKEs the current shape block code into the sprite location register.

Lines 950-960 set up the initial playfield screen. Line 960 is especially important, because if it were not done, the screen would scroll *two* lines at a time each time a fence was moved off the top of the screen.



Lines 970-990 clear the collision registers and set the timer back to 0.

Main Loop. After the set-up subroutines are executed, the program jumps to the main loop. Line 100 reads the keyboard for SHIFT, COMMODORE, and CTRL keypresses. If the end of the game is called for, the program jumps to the End-of-game routine at 300.

Line 105 checks for meaningless keypresses and ignores them. Line 110-115 change the horizontal position, depending on the keypress, and cause the pig to "wrap around" if he goes off the edge of the screen.

Line 120 checks to see if the pig has changed direction. If he has, the sprite shape location register is changed accordingly. Line 125 POKEs the new horizontal position into the two registers affected.

Line 130 checks the timer to see if it's time to scroll the screen. Line 135 checks to see if the player has moved the pig so it bumps into a fence.

Ending Routines. Lines 300-315 display the end of game messages, indicating the score, which gives points for the number of fences displayed before the end and how low the pig was on the screen at the end. Then they move the pig down to the middle of the screen. Line 320 checks for a keypress; if Q is pressed, the program jumps to 370, the end-of-game routine.

Lines 325-330 cause the pig to move along the screen continuously while the program waits for the player to input a choice. Line 345 keeps the program looping through the pig's movement unless the player has pressed P to play again, in which case line 350 resets the initial variables and goes back to line 100 for the next game.

Lines 370-390 put video memory back to normal and reenable all the key combinations that were disabled before; then the program ENDs.

Collision Handling. This routine is only executed if a collision has already taken place. It bounces the pig up to the next valid vertical position. Then it clears the collision registers and checks again to see if the pig is still touching a fence, in which case it starts over again, bouncing the pig up still one more row. If the vertical position, VP, is ever higher than the top of the screen, EG is set to 1, which will end the game when the main loop is executed again.

Scrolling Routines. If we're between fences (RC<DS), then line 502 PRINTs enough cursor-down characters to cause the screen to scroll up one line. The fence-adding routine is skipped.

Lines 506-508 select a new fence segment, PRINT it, and then advance all the counters, changing the difficulty level if necessary. Line 510 checks to see if the maximum number of rows has been added. Right now the game is set up to PRINT 50 new rows on the screen before it ends.

Line 520 checks for collisions. Line 580 clears the screen line link table so the screen will only scroll one line at a time. Then the timer is reset. □

SEE PROGRAM LISTINGS ON PAGE 95

ADDRESSING THE COMMODORE, PART II

The Second of Two Columns on Assembly Language Addressing

By Mark Andrews

Looking for a byte of data in a computer is something like looking for a street address in a large city. If you don't know the city and don't have a map, the task is virtually impossible. If you can find a map, it helps. And with or without a map, the job gets easier as you learn your way around.

Last month, we saw how a set of programming tools called *addressing modes* can be used to retrieve data from memory (and store data in memory) in Commodore 64/Commodore 128 assembly language. In case you missed that column, here's a brief review:

There are 13 addressing modes in C-64/C-128 assembly language. Of these, one of the simplest (and most commonly used) is *absolute addressing*. Using the absolute addressing mode is like going straight to a street address that someone has given you, and immediately finding what you're looking for.

This is the format used for writing a statement using absolute addressing in C-64/C-128 assembly language:

```
LDA $1234
```

The above statement means, "Load the accumulator with the content of memory address $1234." If this statement were encountered in an assembly language program, the contents of the Memory Address $1234 would be loaded into the 6510/8502 *accumulator*, the main internal register in the main microprocessor of a C-64 or a C-128. (As you may recall from previous columns, loading a value into the 6510/8502 accumulator is the initial step in many different kinds of assembly language operations. As you may also remember, the "$" sign in front of the number 1234 means that it's a hexadecimal number.)

INDEXED ADDRESSING

A slightly more complex addressing mode is called *indexed addressing*. Using it is like using a street address to find an apartment building, and then using an apartment number to find an apartment in that building. In a statement written using indexed addressing, the instruction specifies both an address and an *index register* (either the X register or the Y register, if you're programming in C-64/C-128 assembly language). The value of the index register is then added to the specified address, and the sum of this addition operation is the final address of the instruction.

Indexed addressing is often used in loops that are designed to retrieve bytes of data in succession from data tables. Since the X and Y registers are often used as counters in such loops, they have been provided with a special set of instructions that enable them to be incremented and decremented very easily. There are also instructions that can be used to compare the values of the X and Y registers with other values. These comparison instructions are often quite useful in determining the boundaries of data tables.

In C-64/C-128 assembly language, this is the format for writing a statement using indexed addressing:

```
LDA $1234,X
```

A PROGRAM ILLUSTRATING INDEXED ADDRESSING

The Quest, found on page 100 of the program listings section, illustrates the technique of indexed addressing. The program was written using the *Commodore 64 Macro Assembler*, but can be easily modified to work with other assemblers.

One feature of *The Quest* we have not encountered until now is a *directive*, or *pseudo-op*, in Lines 110 to 130.This directive is written ".BYTE" in programs created with the *Commodore 64 Assembler*. If you own a *Merlin 64*, use the *Merlin* equivalent of this directive, which is DFB. If you own a *Panther C-64* the equivalent is DFC.

The .BYTE/DFB/DFC directive is sometimes called a *pseudo-operation code*, or *pseudo-op*, because it appears in the op-code column of assembly language source code listing but is not actually a part of the 6510/8502 assembly language instruction set. Instead, it's a specialized directive that varies in format from assembler to assembler. Many other pseudo-ops have formats that differ from one assembler to another, because there are no generally accepted standards for writing pseudo-op directives.

When the ".BYTE" directive (or one of its equiv-

alents) is used in a program, the bytes that follow are assembled into consecutive locations in RAM. In *The Quest*, the bytes that follow the label TEXT are ASCII codes for a series of text characters– as you will see when you type and run the program.

In *The Quest*, the X register is used as an index register. In Line 150, the X register is loaded with a zero. Then in Line 170, indirect addressing is used to load the accumulator with the first byte of the data segment labeled TEXT, plus the value of the X register. Since the value of the X register starts out as zero, the first value that will be loaded into the accumulator is the first byte of the line labeled TEXT, which is the number 87–or the ASCII code for the letter "W". So, in Line 180, the routine CHROUT (a screen-printing subroutine built into the C-64/C-128 operating system, or *Kernal*), will print a "W" on the screen.

Next, in Line 190, the instruction INX ("increment the X register") is used to increment the value of the X register. Then in Line 200, the value of the X register is compared with the number 23–the length, in bytes, of the text string being printed on the screen. If 23 characters have not yet been printed, the program cycles back to Line 170, labeled LOOP, and the next character is printed. Finally, when all 23 characters have been printed, the program ends.

INDIRECT ADDRESSING

Indirect addressing is a form of indexed addressing slightly more complex than the simple, no-frills indexed addressing used in *The Quest*. When indirect addressing is used in an assembly language program, the *contents* of a pair of memory addresses –rather than the addresses themselves–are added to the value of an index register. The sum is then used as the final address of an instruction.

Indirect addressing could be compared to a situation in which several persons need to get to a meeting each week, but are never sure where the meeting is being held because its location changes from time to time. One solution would be to direct everyone to the same address every week, and let them know in advance where to look when they arrived–under the doormat, for example–for a note containing the address of that week's meeting.

In 6510/8502 assembly language, indirect addressing is often used when rapid changes must be made in data being accessed by a program. If indirect addressing did not exist, huge blocks of data would often have to be moved into and out of programs in real time, resulting in tremendous delays in processing. Fortunately, the existence of indirect addressing can usually eliminate this problem. In a program that uses indirect addressing, it is not usually necessary to incorporate large blocks of data into the body of a program, and to move them around in memory when one block of data must be exchanged with another. Instead, the *address* of a block of data can be used in a program. Then, when it is time to replace that block of data with another block, all a programmer has to do is substitute the *address* of the new block of data for the address of the old one. In this way, by changing only a few bytes of memory during the execution of a program, an assembly language programmer can juggle blocks of data that are hundreds, or even thousands, of bytes long in a tiny fraction of the time it would take to move one whole block of data out of a section of memory and then move another block of data in.

Actually, there are two indirect-addressing modes in 6510/8502 assembly language: *indexed indirect addressing* and *indirect indexed addressing*. As mentioned last month, one way to keep these two confusing terms straight is to remember that the word "indexed," which contains an "X," is the first word in the term "indexed indirect addressing"–the form that uses the X register. And the other mode–indirect indexed addressing–is the mode that makes use of the Y register.

Both indexed indirect addressing and indirect indexed addressing have one unfortunate limitation: they can be used only with zero-page addresses, that is, with memory locations that have addresses ranging from $OO to $FF (or from 0 to 255 in decimal notation). Much more will be said about zero-page addresses in later columns, but this is all we need to know about them right now: since there are only 256 such addresses in the memory space of a Commodore computer–and since most of those are used by the Commodore's BASIC interpreter and operating system–the zero-page memory requirements of

indirect addressing place rather severe restrictions on the use of indirect addressing in user-written programs. These restrictions are especially severe in the case of *indexed indirect* addressing, because indexed indirect addressing can be used only with address tables that are located in their entirety on Page Zero. Because of this limitation, indexed indirect addressing is not often employed in user-written programs. It is a good idea to know how it works, however, since it is sometimes encountered in ROM routines, and can even come in handy now and then in user-written programs, provided that the necessary space on Page Zero can be found.

INDEXED INDIRECT ADDRESSING

When indexed indirect addressing is used in a program, the format is:

```
LDA ($FB,X)
```

If the above statement appeared in an assembly language program, the value of the X register would be added to the value of $FB. The resulting sum would then be interpreted as another address. However, the accumulator would not be loaded with the value of that address. Instead, the *content* of that address, and the content of the *following* address, would be interpreted as still *another* address. In other words, the indexed address would be used as a *pointer* to still another address. And that would be the final address of the instruction LDA.

As an illustration of this concept, let us assume that the X register contained the value 2 when the statement "LDA ($FB,X)" was encountered in a program. First, the number $02—the value of X register—would be added to the value $FB. The sum of this calculation would, of course, be $FD. And the *content* of that address—plus the content of the following address, $FE—would point to the final address to be accessed by the mnemonic LDA.

Now let us suppose the 16-bit value of $1234 were stored in the zero-page pointers $FD and $FE. Since the low byte of a 16-bit value always comes first in 6510/8502 assembly language, the contents of page-zero pointers $FD and $FE would thus be as follows:

ADDRESS	CONTENT
$FD	$34
$FE	$12

Now let us suppose that the value $C000 were stored in memory address $1234. If the value of $1234 were stored in page-zero pointers $FD and $FE, as illustrated above, the 16-bit value contained in $FD and $FE would contain, or *point to,* the address $1234. So the final ef-

fect of the statement "LDA ($FB,X)" would be to load the accumulator with the value $C000.

INDIRECT INDEXED ADDRESSING

Indirect indexed addressing is more versatile—and therefore more useful—than is indexed indirect addressing. When indirect indexed addressing is used to retrieve a value from a table or a block of data, only two bytes of valuable zero-page address space are needed. (In contrast, in the indexed indirect addressing mode, a whole table of address pointers must reside on Page Zero.) Therefore, indirect indexed addressing is used much more often in user-written programs than is indexed addressing.

The format of a statement that uses indirect indexed addressing is:

```
LDA ($FB),Y
```

In the above statement, it is worthwhile to note that while parentheses enclose the address $FB, the "Y" which represents the Y register lies outside the parentheses. This is important to remember, since it means that in indirect indexed addressing, the value of the Y register is used not to index the address $FB, but rather to index the *content* of the address $FB.

Now let us return to the statement "LDA ($FB),Y". If this statement were encountered in an assembly language program, the *content* of memory address $FB, plus the *content* of the following memory address—that is, $FC—would be interpreted as a memory address. Next, the value of the Y register would be added to that address. And finally, the accumulator would be loaded with the *content* of that final address. In other words, the 16-bit value stored in the addresses $FB and $FC would be used as a *pointer* to indicate the final address of the instruction LDA.

To more clearly illustrate how indirect indexed addressing works, let us assume that memory addresses $FB and $FC held the address $1234 when the statement "LDA ($FB),Y was encountered. Since the low byte of a 16-bit value always comes first in 6510/8502 assembly language, the contents of memory addresses $FB and $FC would thus be as follows:

ADDRESS	CONTENT
$FB	$34
$FC	$12

Now let us suppose that the Y register contained a 2 when the statement "LDA ($FB),Y" appeared. In this event, the number $02—the value of the Y register—would be added to the 16-bit address $1234. The sum of this calculation would, of course, be $1236. So the content of memory address $1236 would be interpreted as the final address to be accessed by the mnemonic LDA.

Finally, let us suppose that the value $F0AC were stored in memory address $1236. If this were the case, the final effect of the statement "LDA ($FB),Y" would be to load the accumulator with the value $F0AC.

IMPROVING *THE QUEST*

Before I sign off this month, I'd like to direct your attention to another program, *Response,* which appears after *The Quest* in the program listings section of this issue (see page 100). *Response* works somewhat like *The Quest*, but demonstrates a few more programming techniques—and, in addition, provides an answer to the riddle presented in *The Quest.*

In Line 130, insted of using ASCII codes to represent typed characters, *Response* uses a pair of delimiters to enclose a real string of typed characters—a feature provided in one form or another by most assemblers, including the *Commodore 64 Assembler*, the *Merlin 64*, and the *Panther C-64.*

At the end of this text string, however, there is one ASCII code: the number 13, which equates to an end-of-line (or "EOL") character. (An ASCII code had to be used here, since typing an actual return character would cause the assembler to move on to another line.)

In Lines 150 through 210 of *Response,* a text buffer (created in Line 510) is filled, or "stuffed," with blank characters, or spaces. Next, in Lines 250 through 320, an X loop is used to place the text string in Line 130 in the now-empty text buffer. Then, in Lines 360 through 460, the message in the buffer is printed on the screen.

The loop that prints the message works somewhat differently than the printing loop used in *The Quest;* instead of counting the number of characters in the text string which is being used, this loop keeps checking the accumulator for the presence of an end-of-line character. If no EOL character is detected, the program loops back to Line 370, and prints the next letter on the screen. When the EOL character that ends the text string is finally found, the program ends.

WHAT'S AHEAD

Next month's column will include the most advanced type-and-run program that *has* been presented in this space so far. Called *The Name Game,* it is an interactive game that will demonstrate the use of indirect indexed addressing, and do a bit more, too. *The Name Game*, which I have used for years to demonstrate certain programming principles, is a simple interactive game in which the player converses with the computer. It will not only print messages on the screen, it will also accept typed-in text, and print that on the screen, too. It will then compare the data that is typed in with a string of data stored in a buffer—and will use this comparison to determine the outcome of the game. So—see you in next month's column. □

SEE PROGRAM LISTINGS ON PAGE 100

CADET'S COLUMN

FOR BEGINNING USERS OF THE COMMODORE COMPUTERS – By Cheryl Peterson

Commodore's sales figures for Christmas 1984 showed twice as many disk drive sales as Commodore 64 sales. So it's a safe bet that most of you have already upgraded your systems to include this much-needed peripheral. That's great, because most of this month's column is going to concern the disk drive. For those of you who don't have drives, read this installment to find out some of the reasons for getting one. With the prices being dropped (I've seen them as low as $159), now may be the time to buy.

Despite complaints about the 1541's speed, it's a vast improvement over tape. Having used a disk drive from the very beginning, I can't even contemplate waiting for a tape program to load. Only twice have I been in a situation desperate enough to warrant using the cassette drive, and both times I was elated to go back to my disk.

DISK BASICS

Although they may seem obvious to experienced disk users, there are some basic facts about using a disk based system that you should know. And don't think just because you've been around, you know everything.

Case in point: a professional musician who uses an electronic synthesizer calls my husband (who repairs electronic equipment). Since it's fairly late in the evening, my husband is reluctant to run across town. The problem is eventually traced to the disk drive in the synthesizer. In a brilliant flash of insight my husband asks, "Do you have the drive sitting on a speaker cabinet?" He did! Although moving the drive did not entirely solve the problem, the incident does demonstrate that even experienced users sometimes forget the basics.

Disks (and to a much lesser extent, drives) are susceptible to magnetic fields. A computer diskette is nothing more than a round recording tape; a piece of plastic with an iron oxide coating melded to it. Data is stored on the disk by polarizing the magnetic fields of the iron ions. If you take a disk and place it in an area where there is a high magnetic field (for instance, around speakers connected to stereo systems), it is entirely possible that you will erase the disk or scramble the data.

The plastic on which the iron is mounted is similar to that used in record albums. Heat can warp the plastic and cold can make it brittle. At sufficiently low temperatures, the iron oxide loses its bond to the plastic and there goes your data.

Disk sleeves were made to be used. Please don't throw them away. The keeper of our user group library never could see any reason to take up extra space in files by leaving the jackets on the disks. After about six months of rubbing together, the oxide started falling off, and the entire library (some 120 disks worth) had to be recopied onto new disks. Besides the expense of buying more disks, it took days to make the new copies.

What I'm trying to point out is that disks are fragile. Even the best disks will wear out eventually. But if you treat them right, they can last for several years. Disks that see heavy use may only last a year, but that's why you always keep backup disks, right?!

A good rule of thumb is to make a backup copy of the file in which you are working at least once every half hour. This way you never lose more than 30 minutes of work. Frequently, the backup copy is on the same disk as the original. Most word processors and other heavy use software have a save and continue working feature that will quickly create a duplicate of the file so far and then let you continue where you left off. This is your first backup.

At day's end, it's a good practice to use a copy program to make a duplicate of each disk modified that day. (These copies should be made on disks that will be stored separately from those you use each day.) If you are doing anything more important than writing letters to goldfish, a safety is a necessity. I hope to compare a few of the commercial copy programs in a future column. *Fast Load* has such a program built into it.

ORGANIZATION

Most beginners don't have any idea how fast the disks collect. With a typical family, before you know it, you'll have disks with programs written in BASIC, COMAL, PASCAL, machine language, or any combination thereof. You'll have databases, letters created with different word processors, songs written with music processors, sprites made with sprite generators, pictures created with a variety of graphics packages, and data disks that track your progress with the very latest in adventure games.

And all of them will have cryptic labels that list about three files! If you're lucky, there might even be a semi-meaningful diskname. Something like WN!DD1.01 (*Write Now!* Data Disk 1.01) And if you've been making safeties like you should, whew!

Now for the good news! You, as beginners, are in the perfect position to nip this problem in the bud. Start out with some plan for labeling disks. Everyone who uses the computer should help come up with a logical system for organizing disks. I recommend using disknames that key the data to whatever disk was used to create it and then a general category type.

For instance:

WN!Personal Let.JA
WN!Personal Let.FB
WN!Business Let.01

BSWTom Creative.WR
BSWSusan Book R.01
BSWSusan Book R.02
Ahoy! PRG Files.01
CompuServe Down.01
Cher PRG Games.01

With a little imagination, these disknames can identify the disk's information; WN!=*Write Now!*, Let=Letters, JA=January, FB=February, BSW=*Bank Street Writer*, WR=Writing, R=Reports, PRG=program files, Down=Downloaded files. Tom, Susan, and Cher are family members who regularly use the disk.

It took me three days to organize the mess that six months of haphazard disk and file naming had created. Oh, how I wished someone had warned me to be organized from the beginning.

Even with organization, it can still be tough to find a particular file that you haven't looked at for months. A good disk catalog program will really help here. I've seen one that costs about $30 and includes an option to print labels.

MACH 5's DISK CATALOG PROGRAM

For my money, I'd get Access Software's *Mach 5* cartridge instead. Although it doesn't print labels, its features are almost indispensible. Besides speeding up disk access, *Mach 5* enables a catalog program that comes on disk.

And is it a dandy! After creating a data disk for your catalog to reside on, you just swap in a disk whose directory you want added to the catalog. Take that disk out and put the catalog data disk back in. That's about it.

Each disk that you want added is swapped in. When you get them all entered, a sort routine in the program will alphabetize them. The printout looks great! (see figure 1). Not only do you know where to find the file you're looking for, you can see how big it is. I usually keep a printout of the catalog in a binder, next to my disk files.

Another option lets you view or print a listing of the disks in a catalog (see figure 2). This is especially useful at the beginning of your catalog, since it lists how much free space each disk has. Ever wonder which of your disks has enough room for a 20 block file and spend half an hour swapping disks in and out to find one? With this program, you never will.

To update a listing, you delete modified disks by entering their disknames. It isn't necessary to delete each file name; the program automatically finds each one associated with a deleted diskname and removes it. Then you use the "Add a disk function" to update the files.

There's even an option to change a diskname without having to reformat the whole disk (something I've seen no other program do). Even if you're a few months along in "haphazard" mode, it isn't too late to start getting organized with *Mach 5*.

One big hassle facing new drive users is learning the new, save, load, format, initialize, and validate commands. A disk provided with the drives contains a DOS (Disk Operating System) wedge that is supposed to make using these features easier. The wedge must be loaded before using it, however. *Mach 5* has a built in DOS wedge, accessible from BASIC. Because it is on cartridge, though, *Mach 5*'s DOS doesn't have to be loaded from BASIC. The command structure looks much like that of the Commodore DOS wedge, so the commands aren't confusing to more experienced users.

Beginners may not be aware that before you can use

Page 1

File Name	File Type	File Size	Disk Name	Starting Track	Starting Sector	Entry Order
,	SEQ	1	fourth	24	3	4
....	PRG	1	fifth	11	0	5
21ift	SEQ	3	fifth	19	2	5
=	PRG	1	write now!	20	7	6
@	PRG	1	first	23	18	2
ahoy!.bak	SEQ	1	mj data	20	18	11
ahoy!.rev	SEQ	1	mj data	20	17	11
alphaloop.o	PRG	1	assembly	16	4	9
alphaloop.s	PRG	1	assembly	16	3	9

FIGURE 1 **(top): a *Mach 5*-generated disk directory.**
FIGURE 2 **(right): list of disks in *Mach 5* catalog.**

List of Directories

1	ledger data file	Free Sectors = 645
2	first	Free Sectors = 423
3	second	Free Sectors = 574
4	fourth	Free Sectors = 333
5	fifth	Free Sectors = 197
6	write now!	Free Sectors = 517
7	ok8	Free Sectors = 635
8	ok7	Free Sectors = 632
9	assembly	Free Sectors = 604
10	data	Free Sectors = 641
11	mj data	Free Sectors = 508

a disk with the Commodore drive, it must be formatted. This two-minute process requires typing the command line (in BASIC):

```
PRINT#15,"NEW0:NAME,ID"
```

where "name,id" is the diskname and identification code. If the DOS wedge is loaded or the *Mach 5* cartridge is in, you can type @ instead of PRINT#15.

Even without its quick loading features, *Mach 5* is an excellent package. And at $34.95, it's a bargain, too.

TIPS ON BUYING SOFTWARE

Although I don't buy software often, there are a few tricks I've picked up along the way that I'll pass on to you. (Don't be surprised if they don't agree with anything you've heard elsewhere.)

Before deciding to buy any "productivity" type software, check back issues of magazines for buyer's guides and indepth reviews. Make a list of the features you definitely need and those you'd like to have if the price isn't too high. Be careful not to pay exorbitant prices for "bells and whistles" you don't need or want.

Read as many current Commodore magazines as you can get your hands on. Don't be an indiscriminate reader; look for useful buying information. Read magazines with a pen in your hand and a pile of note cards sitting next to you. If the family's decided to buy a word processing program in the next couple of weeks, make notes on all the pertinent advertisements: software name and manufacturer, what features are offered, asking price, page and magazine of advertisement. If you see a package that interests you, look for the discounted price being offered by mail order firms.

Mind you, I didn't recommend buying it from a mail order company. Find a local dealer who doesn't mind letting you hang around the store. Hang around. Try out the software you're interested in. Try it on a system similar to yours. Be sure the package will work with your hardware combinations. What kind of printer, interface, modem, or input devices do you need? Does that software need a high resolution or low resolution light pen?

While you're in the store, check the prices of disks. Even if you don't find the software you want, you can always use a few more disks and the store owner may be running a sale. You'd be surprised at the number of times I've walked out of a store empty-handed only to discover two days later that I had only one clean disk left.

Before buying, compare prices *realistically.*

Most mail order places now add a three to four per cent surcharge to credit card orders or a three to five dollar charge for COD. On top of that, there's a three dollar to five per cent shipping and handling charge. By the time you add it all up, you may only be saving a couple of dollars.

Then there's the waiting. If you buy software at a local store, it's yours when you walk out. You run right home and use it. Not so with mail order. If you pay by check, there's a two week wait while it clears. Discounters advertise some products that are not in stock, since magazine ads must be placed two to three months in advance. A particularly popular package may be temporarily sold out, as well. Even if it is in stock, UPS generally takes 5 to 8 days to deliver.

Though mail order companies must ship your order within 30 days or inform you of the delay and offer to refund your money, they are not required to pay you interest on your money while they hold it. If there's a chance the software will come in a future shipment, they'll usually advise you to wait another couple of weeks before requesting a refund. After all, it takes the accounting department a couple of weeks to issue a refund check.

For your two dollar savings, you've given away the opportunity to take the software back to the store for a credit if it doesn't work. Since most local dealers want to keep your business, they'll try very hard to make you happy. Usually all you need do is bring the software back and show that it doesn't work on their computers, either.

This may lead to finding you actually have a hardware problem, or it could get you another copy of the software. At the very least, if the software won't load, most stores will give you a credit against any other merchandise you want to buy. Refunds are almost unheard of, no matter where you buy.

If you bought from a discount house, you can be almost guaranteed hassles. Don't hold your breath while waiting for most mail order houses to issue a return authorization number. Did I mention that you'll have to call on their user support line, so you get to pay long distance charges? 800 numbers are for orders only. When you do get permission to send it back, you go stand in line at the post office to send the package "Return receipt requested." This way you get back a post card that confirms that software company "XYZ" got the software.

And wait, and wait, and wait.

After several weeks, you may get a package with another copy of the software. If not, you call the company and hassle it out.

With certain software manufacturers, this isn't necessary. They offer a warranty! You ship the faulty disks to their main office and they send you new ones that should work. Usually, it'll only cost you another five dollars more plus shipping. Don't forget "Return receipt requested." And should the problem be hardware related, the new disks won't work, either.

So don't be so sure mail order is really cheaper. It may save a couple of dollars initially, but you'll probably pay for it in waiting time later.

Buying software can be a harrowing exercise. The best protection against disappointment is research. Magazines are a good starting point, but personal recommendations from friends or members of the local user group can lead to bargains. Try before you buy is still the best guarantee of satisfaction, though.

Next month, a few words about fans, furniture, and other fun stuff. □

PROGRAM LISTINGS

Attention new *Ahoy!* readers! You must read the following information very carefully prior to typing in programs listed in *Ahoy!* Certain Commodore characters, commands, and strings of characters and commands will appear in a special format. Follow the instructions and listing guide on this page.

On the following pages you'll find several programs that you can enter on your Commodore computer. But before doing so, read this entire page carefully.

To insure clear reproductions, *Ahoy!*'s program listings are generated on a daisy wheel printer, incapable of printing the commands and graphic characters used in Commodore programs. These are therefore represented by various codes enclosed in brackets [ ]. For example: the SHIFT CLR/HOME command is represented onscreen by a heart . The code we use in our listings is [CLEAR]. The chart below lists all such codes which you'll encounter in our listings, except for one other special case.

The other special case is the COMMODORE and SHIFT characters. On the front of most keys are two symbols. The symbol on the left is obtained by pressing that key while holding down the COMMODORE key; the symbol on the right, by pressing that key while holding down the SHIFT key. COMMODORE and SHIFT characters are represented in our listings by a lower-case "s" or "c" followed by the symbol of the key you must hit. COMMODORE J, for example, is represented by [c J], and SHIFT J by [s J].

Additionally, any character that occurs more than two times in a row will be displayed by a coded listing. For example, [3 "[LEFT]"] would be 3 CuRSoR left commands in a row, [5 "[s EP]"] would be 5 SHIFTed English Pounds, and so on. Multiple blank spaces will be noted in similar fashion: e.g., 22 spaces as [22 " "].

Sometimes you'll find a program line that's too long for the computer to accept (C-64 lines are a maximum of 80 characters, or 2 screen lines long; VIC 20 lines, a maximum of 88 characters, or 4 screen lines). To enter these lines, refer to the *BASIC Command Abbreviations Appendix* in your User Manual.

On the next page you'll find our *Bug Repellent* programs for the VIC 20 and C-64. The version appropriate for your machine will help you proofread our programs after you type them. (Please note: the *Bug Repellent* line codes that follow each program line, in the whited-out area, should *not* be typed in. See the instructions preceding each program.)

Also on the following page you will find *Flankspeed*, our ML entry program, and instructions on its use. □

Call *Ahoy!* at 212-239-0855 with any problems.

When You See	It Means	You Type		You Will See
[CLEAR]	Screen Clear	SHIFT	CLR/HOME	
[HOME]	Home		CLR/HOME	
[UP]	Cursor Up	SHIFT	↑CRSR↓	
[DOWN]	Cursor Down		↑CRSR↓	
[LEFT]	Cursor Left	SHIFT	←CRSR→	
[RIGHT]	Cursor Right		←CRSR→	
[SS]	Shifted Space	SHIFT	Space	
[INSERT]	Insert	SHIFT	INST/DEL	
[DEL]	Delete		INST/DEL	
[RVSON]	Reverse On	CNTRL	9	
[RVSOFF]	Reverse Off	CNTRL	0	
[UPARROW]	Up Arrow		↑	
[BACKARROW]	Back Arrow		←	
[PI]	PI		π	
[EP]	English Pound		£	

When You See	It Means	You Type		You Will See
[BLACK]	Black	CNTRL	1	
[WHITE]	White	CNTRL	2	
[RED]	Red	CNTRL	3	
[CYAN]	Cyan	CNTRL	4	
[PURPLE]	Purple	CNTRL	5	
[GREEN]	Green	CNTRL	6	
[BLUE]	Blue	CNTRL	7	
[YELLOW]	Yellow	CNTRL	8	
[F1]	Function 1		F1	
[F2]	Function 2	SHIFT	F1	
[F3]	Function 3		F3	
[F4]	Function 4	SHIFT	F3	
[F5]	Function 5		F5	
[F6]	Function 6	SHIFT	F5	
[F7]	Function 7		F7	
[F8]	Function 8	SHIFT	F7	

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** This and the preceding explain these codes and provide other essential information on entering **Ahoy!** programs. Read these pages **before** entering programs.

BUG REPELLENT

This program will let you debug any *Ahoy!* program. Follow instructions for VIC 20 (cassette or disk) or C-64.

VIC 20 VERSION

By Michael Kleinert and David Barron

For cassette: type in and save the *Bug Repellent* program, then type RUN 63000[RETURN]SYS 828[RETURN]. If you typed the program properly, it will generate a set of two-letter line codes that will match those listed to the right of the respective program lines.

Once you've got a working *Bug Repellent*, type in the program you wish to check. Save it and type the RUN and SYS commands listed above once again, then compare the line codes generated to those listed in the magazine. If you spot a discrepancy, a typing error exists in that line. Important: you must use exactly the same spacing as the program in the magazine. Due to memory limitations on the VIC, the VIC *Bug Repellent* will register an error if your spacing varies from what's printed.

You may type SYS 828 as many times as you wish, but if you use the cassette for anything, type RUN 63000 to restore the *Repellent*.

When your program has been disinfected you may delete all lines from 63000 on. (Be sure the program you type doesn't include lines above 63000!)

For disk: enter *Bug Repellent*, save it, and type RUN:NEW [RETURN]. Type in the program you wish to check, then SYS 828.

To pause the line codes listing, press SHIFT.

To send the list to the printer type OPEN 4,4:CMD 4:SYS 828[RETURN]. When the cursor comes back, type PRINT#4:CLOSE 4[RETURN].

```
•63000 FORX=828TO1023:READY:POKEX,Y:NEXT:END    AC
•63001 DATA169,0,133,63,133,64,165,43,133,251   JL
•63002 DATA165,44,133,252,160,0,132,254,32,228 DF
•63003 DATA3,234,177,251,208,3,76,208,3,230    OE
•63004 DATA251,208,2,230,252,169,244,160,3,32  OH
•63005 DATA30,203,160,0,177,251,170,230,251,20
 8                                             KO
•63006 DATA2,230,252,177,251,32,205,221,169,58 JJ
•63007 DATA32,210,255,169,0,133,253,230,254,32 OK
•63008 DATA228,3,234,165,253,160,0,170,177,251 LG
•63009 DATA201,32,240,6,138,113,251,69,254,170 BP
•63010 DATA138,133,253,177,251,208,226,165,253
 ,41                                           DD
•63011 DATA240,74,74,74,74,24,105,65,32,210    EK
•63012 DATA255,165,253,41,15,24,105,65,32,210  FO
•63013 DATA255,169,13,32,210,255,173,141,2,41  PK
•63014 DATA1,208,249,230,63,208,2,230,64,230   CB
•63015 DATA251,208,2,230,252,76,74,3,169,236   KH
•63016 DATA160,3,32,30,203,166,63,165,64,32    DP
•63017 DATA205,221,169,13,32,210,255,96,230,25
 1                                             EL
•63018 DATA208,2,230,252,96,0,76,73,78,69      OI
•63019 DATA83,58,32,0,76,73,78,69,32,35        FG
•63020 DATA32,0,0,0,0,0                        LE
```

C-64 VERSION

By Michael Kleinert and David Barron

Type in, SAVE, and RUN the *Bug Repellent*. Type NEW, then type in or LOAD the *Ahoy!* program you wish to check. When that's done, SAVE your program (don't RUN it!) and type SYS 49152 [RETURN].

To pause the listing depress and hold the SHIFT key.

Compare the codes your machine generates to the codes listed to the right of the respective program lines. If you spot a difference, an error exists in that line. Jot down the number of lines where contradictions occur, LIST each line, spot the errors, and correct them.

```
•5000 FORX=49152TO49488:READY:POKEX,Y:NEXT:END GJ
•5001 DATA32,161,192,165,43,133,251,165,44,133 DL
•5002 DATA252,160,0,132,254,32,37,193,234,177  DB
•5003 DATA251,208,3,76,138,192,230,251,208,2   OF
•5004 DATA230,252,76,43,192,76,73,78,69,32     KN
•5005 DATA35,32,0,169,35,160,192,32,30,171     CA
•5006 DATA160,0,177,251,170,230,251,208,2,230  CE
•5007 DATA252,177,251,32,205,189,169,58,32,210 JE
•5008 DATA255,169,0,133,253,230,254,32,37,193  CL
•5009 DATA234,165,253,160,0,76,13,193,133,253  NB
•5010 DATA177,251,208,237,165,253,41,240,74,74 MB
•5011 DATA74,74,24,105,65,32,210,255,165,253   EP
•5012 DATA41,15,24,105,65,32,210,255,169,13    GH
•5013 DATA32,220,192,230,63,208,2,230,64,230   AN
•5014 DATA251,208,2,230,252,76,11,192,169,153  NG
•5015 DATA160,192,32,30,171,166,63,165,64,76   BF
•5016 DATA231,192,96,76,73,78,69,83,58,32      EP
•5017 DATA0,169,247,160,192,32,30,171,169,3    PJ
•5018 DATA133,254,32,228,255,201,83,240,6,201  FK
•5019 DATA80,208,245,230,254,32,210,255,169,4  FL
•5020 DATA166,254,160,255,32,186,255,169,0,133 CL
•5021 DATA63,133,64,133,2,32,189,255,32,192    GC
•5022 DATA255,166,254,32,201,255,76,73,193,96  NN
•5023 DATA32,210,255,173,141,2,41,1,208,249    NH
•5024 DATA96,32,205,189,169,13,32,210,255,32   IM
•5025 DATA204,255,169,4,76,195,255,147,83,67   KC
•5026 DATA82,69,69,78,32,79,82,32,80,82        DC
•5027 DATA73,78,84,69,82,32,63,32,0,76         ML
•5028 DATA44,193,234,177,251,201,32,240,6,138  GN
•5029 DATA113,251,69,254,170,138,76,88,192,0   JK
•5030 DATA0,0,0,230,251,208,2,230,252,96       NA
•5031 DATA170,177,251,201,34,208,6,165,2,73    DM
•5032 DATA255,133,2,165,2,208,218,177,251,201  JA
•5033 DATA32,208,212,198,254,76,29,193,0,169   FM
•5034 DATA13,76,210,255,0,0,0                  PA
```

FLANKSPEED FOR THE C-64

By Gordon F. Wheat

Flankspeed will allow you to enter machine language *Ahoy!* programs without any mistakes. Once you have typed the program in, save it for future use. While entering an ML program with *Flankspeed* there is no need to enter spaces or hit the carriage return. This is all done automatically. If you make an error in a line a bell will ring and you will be asked to enter it again. To LOAD in a program Saved with *Flankspeed* use LOAD "name",1,1 for tape, or LOAD "name",8,1 for disk. The function keys may be used after the starting and ending addresses have been entered.

f1—SAVEs what you have entered so far.

f3—LOADs in a program worked on previously.

f5—To continue on a line you stopped on after LOADing in the previously saved work.

f7—Scans through the program to locate a particular line, or to find out where you stopped the last time you entered the program. f7 temporarily freezes the output as well.

```
•5 POKE53280,12:POKE53281,11                   LL
•6 PRINT"[CLEAR][c 8][RVSON][15" "]FLANKSPEED[
 15" "]";                                      ED
•10 PRINT"[RVSON][5" "]MISTAKEPROOF ML ENTRY P
 ROGRAM[6" "]"                                 MC
•15 PRINT"[RVSON][9" "]CREATED BY G. F. WHEAT[
 9" "]"                                        DM
•20 PRINT"[RVSON][3" "]COPR. 1984, ION INTERNA
```

```
TIONAL INC.[3" "]"                              DH
30 FORA=54272TO54296:POKEA,0:NEXT               IM
40 POKE54272,4:POKE54273,48:POKE54277,0:POKE5
4278,249:POKE54296,15                           NH
70 FORA=680TO699:READB:POKEA,B:NEXT             KO
75 DATA169,251,166,253,164,254,32,216,255,96    HJ
76 DATA169,0,166,251,164,252,32,213,255,96      JB
80 B$="STARTING ADDRESS IN HEX":GOSUB2010:AD=
B:SR=B                                          HC
85 GOSUB2520:IFB=0THEN80                        FO
86 POKE251,T(4)+T(3)*16:POKE252,T(2)+T(1)*16    KE
90 B$="ENDING ADDRESS IN HEX":GOSUB2010:EN=B    IF
95 GOSUB2510:IFB=0THEN80                        FP
96 POKE254,T(2)+T(1)*16:B=T(4)+1+T(3)*16        MN
97 IFB>255THENB=B-255:POKE254,PEEK(254)+1       GE
98 POKE253,B:PRINT                              HN
100 REM  GET HEX LINE                           IL
110 GOSUB3010:PRINT": [c P][LEFT]";:FORA=0TO8   FG
120 FORB=0TO1:GOTO210                           MD
125 NEXTB                                       ME
130 A%(A)=T(1)+T(0)*16:IFAD+A-1=ENTHEN310       LH
135 PRINT" [c P][LEFT]";                        IK
140 NEXTA:T=AD-(INT(AD/256)*256):PRINT" "       PD
150 FORA=0TO7:T=T+A%(A):IFT>255THENT=T-255      LK
160 NEXT                                        IA
170 IFA%(8)<>TTHENGOSUB1010:GOTO110             FK
180 FORA=0TO7:POKEAD+A,A%(A):NEXT:AD=AD+8:GOT
O110                                            MN
200 REM    GET HEX INPUT                        AB
210 GETA$:IFA$=""THEN210                        HO
211 IFA$=CHR$(20)THEN270                        GC
212 IFA$=CHR$(133)THEN4000                      MD
213 IFA$=CHR$(134)THEN4100                      KF
214 IFA$=CHR$(135)THENPRINT" ":GOTO4500         GE
215 IFA$=CHR$(136)THENPRINT" ":GOTO4700         BJ
220 IFA$>"@"ANDA$<"G"THENT(B)=ASC(A$)-55:GOTO
250                                             GM
230 IFA$>"/"ANDA$<":"THENT(B)=ASC(A$)-48:GOTO
250                                             LE
240 GOSUB1100:GOTO210                           LL
250 PRINTA$"[c P][LEFT]";                       OA
260 GOTO125                                     CG
270 IFA>0THEN280                                OP
272 A=-1:IFB=1THEN290                           OB
274 GOTO140                                     CJ
280 IFB=0THENPRINTCHR$(20);CHR$(20);:A=A-1      HG
285 A=A-1                                       BE
290 PRINTCHR$(20);:GOTO140                      KH
300 REM LAST LINE                               AD
310 PRINT" ":T=AD-(INT(AD/256)*256)             GJ
320 FORB=0TOA-1:T=T+A%(B):IFT>255THENT=T-255    PL
330 NEXT                                        IA
340 IFA%(A)<>TTHENGOSUB1010:GOTO110             KF
350 FORB=0TOA-1:POKEAD+B,A%(B):NEXT             HN
360 PRINT:PRINT"YOU ARE FINISHED!":GOTO4000     ON
1000 REM BELL AND ERROR MESSAGES                FL
1010 PRINT:PRINT"LINE ENTERED INCORRECTLY":PR
INT:GOTO1100                                    DH
1020 PRINT:PRINT"INPUT A 4 DIGIT HEX VALUE!":
GOTO1100                                        JA
1030 PRINT:PRINT"ENDING IS LESS THAN STARTING
!":B=0:GOTO1100                                 HD
1040 PRINT:PRINT"ADDRESS NOT WITHIN SPECIFIED
 RANGE!":B=0:GOTO1100                           AG
1050 PRINT:PRINT"NOT ZERO PAGE OR ROM!":B=0:G
OTO1100                                         KN
1060 PRINT"?ERROR IN SAVE":GOTO1100             EI
1070 PRINT"?ERROR IN LOAD":GOTO1100             GL
1080 PRINT:PRINT:PRINT"END OF ML AREA":PRINT    PG
1100 POKE54276,17:POKE54276,16:RETURN           BH
1200 OPEN15,8,15:INPUT#15,A,A$:CLOSE15:PRINTA
$:RETURN                                        IM
2000 REM GET FOUR DIGIT HEX                     PC
2010 PRINT:PRINTB$;:INPUTT$                     GM
2020 IFLEN(T$)<>4THENGOSUB1020:GOTO2010         II
2040 FORA=1TO4:A$=MID$(T$,A,1):GOSUB2060:IFT(
A)=16THENGOSUB1020:GOTO2010                     AD
2050 NEXT:B=(T(1)*4096)+(T(2)*256)+(T(3)*16)+
T(4):RETURN                                     GF
2060 IFA$>"@"ANDA$<"G"THENT(A)=ASC(A$)-55:RET
URN                                             EH
2070 IFA$>"/"ANDA$<":"THENT(A)=ASC(A$)-48:RET
URN                                             KP
2080 T(A)=16:RETURN                             NP
2500 REM ADRESS CHECK                           LI
2510 IFAD>ENTHEN1030                            MI
2515 IFB<SRORB>ENTHEN1040                       MG
2520 IFB<256OR(B>40960ANDB<49152)ORB>53247THE
N1050                                           MI
2530 RETURN                                     IM
3000 REM ADDRESS TO HEX                         EB
3010 AC=AD:A=4096:GOSUB3070                     HG
3020 A=256:GOSUB3070                            CE
3030 A=16:GOSUB3070                             PN
3040 A=1:GOSUB3070                              MJ
3060 RETURN                                     IM
3070 T=INT(AC/A):IFT>9THENA$=CHR$(T+55):GOTO3
090                                             CJ
3080 A$=CHR$(T+48)                              JP
3090 PRINTA$;:AC=AC-A*T:RETURN                  AC
4000 A$="**SAVE**":GOSUB4200                    AI
4050 OPEN1,T,1,A$:SYS680:CLOSE1                 LH
4060 IFST=0THENEND                              EO
4070 GOSUB1060:IFT=8THENGOSUB1200               FJ
4080 GOTO4000                                   FF
4100 A$="**LOAD**":GOSUB4200                    AB
4150 OPEN1,T,0,A$:SYS690:CLOSE1                 MF
4160 IFST=64THEN110                             JH
4170 GOSUB1070:IFT=8THENGOSUB1200               CM
4180 GOTO4100                                   FO
4200 PRINT" ":PRINTTAB(14)A$                    FG
4210 PRINT:A$="":INPUT"FILENAME";A$             OM
4215 IFA$=""THEN4210                            GF
4220 PRINT:PRINT"TAPE OR DISK?":PRINT           DF
4230 GETB$:T=1:IFB$="D"THENT=8:A$="@0:"+A$:RE
TURN                                            IG
4240 IFB$<>"T"THEN4230                          FN
4250 RETURN                                     IM
4500 B$="CONTINUE FROM ADDRESS":GOSUB2010:AD=
B                                               DK
4510 GOSUB2515:IFB=0THEN4500                    MA
4520 PRINT:GOTO110                              OI
4700 B$="BEGIN SCAN AT ADDRESS":GOSUB2010:AD=
B                                               FH
4705 GOSUB2515:IFB=0THEN4700                    NK
4706 PRINT:GOTO4740                             DI
4710 FORB=0TO7:AC=PEEK(AD+B):GOSUB3030:IFAD+B
=ENTHENAD=SR:GOSUB1080:GOTO110                  BK
4715 PRINT" ";:NEXTB                            EC
4720 PRINT:AD=AD+8                              GN
4730 GETB$:IFB$=CHR$(136)THEN110                MN
4740 GOSUB3010:PRINT": ";:GOTO4710              JD
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

DRAGON TYPE

FROM PAGE 61 — PROGRAM 1

```
•10 REM PROGRAM 1                              NA
•20 PRINT"[CLEAR][DOWN][DOWN] LOADING MAC
 HINE LANGUAGE"                                MP
•30 FORI=49152TO49946:READA:POKEI,A:NEXT:
 REM CHECKSUM=98881                            CI
•40 PRINT"[DOWN] LOADING DRAGON TYPE2"         GP
•50 LOAD"DRAGON TYPE2",8                       HP
•100 DATA32,110,193,169,205,133,128,169        DN
•101 DATA4,133,129,169,11,141,39,208           EH
•102 DATA141,40,208,141,41,208,169,1           NE
•103 DATA141,21,208,169,0,141,25,195           BP
•104 DATA141,0,208,141,16,208,141,26           CH
•105 DATA195,169,85,141,203,194,173,23         BM
•106 DATA195,141,206,194,169,11,141,207        KG
•107 DATA194,32,38,193,169,15,141,248          KK
•108 DATA7,169,0,141,209,194,32,154            PN
•109 DATA192,201,136,240,69,32,38,193          CE
•110 DATA169,0,141,16,208,141,0,208            EH
•111 DATA24,173,203,194,105,16,201,220         AH
•112 DATA176,9,141,203,194,32,132,193          BN
•113 DATA76,57,192,169,14,141,248,7            DB
•114 DATA169,0,141,16,208,169,164,141          OJ
•115 DATA0,208,169,70,141,1,208,169            NM
•116 DATA0,141,39,208,169,5,141,33             NK
•117 DATA208,169,13,141,32,208,32,121          HN
•118 DATA193,96,169,1,141,26,195,76            KF
•119 DATA107,192,32,228,255,201,136,240        KK
•120 DATA55,201,32,144,245,141,208,194         GF
•121 DATA32,217,192,224,1,240,235,201          LI
•122 DATA91,176,231,169,14,141,248,7           KK
•123 DATA32,3,193,32,153,193,174,24            MO
•124 DATA195,32,103,193,169,15,141,248         HA
•125 DATA7,32,3,193,173,16,208,240             NJ
•126 DATA201,173,0,208,201,28,144,194          MJ
•127 DATA96,172,209,194,177,128,201,32         CF
•128 DATA144,27,56,237,208,194,240,11          LA
•129 DATA238,25,195,32,197,193,162,1           AC
•130 DATA76,252,192,169,32,145,128,238         JO
•131 DATA209,194,162,0,96,24,105,64            ON
•132 DATA76,226,192,169,4,141,204,194          GE
•133 DATA238,0,208,173,0,208,141,0             DD
•134 DATA208,208,5,169,1,141,16,208            NP
•135 DATA162,2,32,103,193,206,204,194          FK
•136 DATA173,204,194,208,227,96,173,203        EB
•137 DATA194,141,1,208,173,207,194,141         HA
•138 DATA205,194,169,4,141,204,194,238         FH
•139 DATA0,208,174,206,194,32,103,193          EN
•140 DATA206,204,194,173,204,194,208,239       AJ
•141 DATA173,248,7,201,13,208,6,238            GA
•142 DATA248,7,238,248,7,206,248,7             LL
•143 DATA206,205,194,173,205,194,208,210       AO
•144 DATA96,160,255,136,208,253,96,32          ID
•145 DATA97,193,202,208,250,96,165,128         JD
•146 DATA141,211,194,165,129,141,210,194       ND
•147 DATA96,173,211,194,133,128,173,210        NE
•148 DATA194,133,129,96,24,177,128,32          AA
•149 DATA210,255,105,64,165,128,105,80         JF
•150 DATA133,128,165,129,105,0,133,129         GC
•151 DATA96,169,15,141,0,212,169,85            OB
•152 DATA141,24,212,169,18,141,1,212           KG
•153 DATA169,65,141,6,212,169,28,141           IE
•154 DATA5,212,169,129,141,4,212,162           GA
•155 DATA22,32,97,193,202,208,250,169          AA
•156 DATA128,141,4,212,96,169,0,141            EN
•157 DATA0,212,169,2,141,32,208,169            IG
•158 DATA85,141,24,212,169,38,141,1            FA
•159 DATA212,169,65,141,6,212,169,45           MK
•160 DATA141,5,212,169,33,141,4,212            BF
•161 DATA162,45,32,97,193,202,208,250          LM
•162 DATA169,32,141,4,212,169,5,141            KB
•163 DATA32,208,96,160,255,169,3,141           LA
•164 DATA33,208,169,14,141,32,208,200          PL
•165 DATA185,212,194,32,210,255,201,0          PN
•166 DATA208,245,169,163,160,40,32,210         LL
•167 DATA255,136,208,250,32,110,193,162        MK
•168 DATA200,32,103,193,169,157,133,128        FH
•169 DATA169,5,133,129,169,0,141,16            JN
•170 DATA208,169,3,141,0,208,169,11            EN
•171 DATA141,39,208,141,40,208,141,41          LG
•172 DATA208,169,8,141,25,195,169,125          EP
•173 DATA141,203,194,169,26,141,207,194        JC
•174 DATA169,25,141,206,194,169,0,141          OO
•175 DATA209,194,162,255,32,103,193,206        LO
•176 DATA25,195,173,25,195,208,243,169         EP
•177 DATA1,141,21,208,32,38,193,169            KF
•178 DATA15,141,248,7,162,180,32,103           JI
•179 DATA193,169,14,141,248,7,32,153           OH
•180 DATA193,172,209,194,32,243,192,32         LI
•181 DATA3,193,162,55,32,103,193,169           DH
•182 DATA15,141,248,7,32,3,193,162             GG
•183 DATA80,32,103,193,173,209,194,201         NO
•184 DATA13,208,214,169,13,141,248,7           GK
•185 DATA162,100,32,103,193,169,8,141          IB
•186 DATA206,194,169,11,141,207,194,32         FN
•187 DATA38,193,169,1,141,16,208,169           EA
•188 DATA27,141,207,194,32,38,193,32           FO
•189 DATA121,193,96,125,0,0,8,27               CH
•190 DATA0,13,32,201,147,28,29,29              PB
•191 DATA29,29,29,29,29,29,29,29               NO
•192 DATA29,29,29,29,17,17,17,17               KC
•193 DATA17,17,17,17,68,82,65,71               EF
•194 DATA79,78,32,84,89,80,69,13               PK
•195 DATA13,29,29,29,29,29,29,29               FH
•196 DATA29,29,29,29,29,29,66,89               PH
•197 DATA32,66,79,66,32,83,80,73               JD
•198 DATA82,75,79,31,13,13,0,1                 AC
•199 DATA10,0,0                                JF
```

PROGRAM 2

```
•10 REM PROGRAM 2                              NH
•20 PRINT"[DOWN] RUN"                          PC
•30 N=99:DIMW$(N):SC=53280                     DJ
•100 FORJ=832TO1022:READB:POKEJ,B:NEXT         KH
•130 POKE2040,13:POKE2041,14:POKE2042,15       ND
•150 POKE792,193:SYS49659                      CK
•155 FORI=0TON:READW$(I):NEXT                  CN
•160 PRINT"[HOME][15"[DOWN]"][BLACK]"TAB(
 10)"CHOOSE TYPING SPEED[DOWN]"                JJ
•170 PRINTTAB(15)"[BLUE]1. FAST[DOWN]":PR
 INTTAB(15)"2. MEDIUM[DOWN]":PRINTTAB(15)
 "3. SLOW"                                     LK
•190 GETA$:IFA$=""THEN190                      HJ
•200 A=VAL(A$):IFA<1ORA>3THEN190               DP
•210 POKE49943,A:POKE49944,A*30                KB
•500 PRINT"[CLEAR][4"[DOWN]"][BLUE]":POKE
 SC,5:POKESC+1,15                              DH
•510 FORJ=1TO9:A$="[4" "]":FORI=0TO6:R=IN
 T(RND(0)*N)                                   EL
•520 A$=A$+" "+W$(R):NEXT                      DN
•530 PRINTLEFT$(A$,35):PRINT:NEXT:POKE198
 ,0:T0=TI:SYS49152                             NP
•540 T=(TI-T0)/3600:E=PEEK(49945):POKESC,
 5:POKESC+1,13:S=INT(47/T+.5)                  NI
•545 IFPEEK(49946)THENS=0:E=0                  CH
•550 SR=S*90-20*E:IFSR>HSTHENHS=SR             NI
•560 S$=RIGHT$(STR$(S),2):E$=RIGHT$(STR$(
 E),2)                                         CG
•565 SR$=RIGHT$(STR$(SR),4):HS$=RIGHT$(ST
 R$(HS),4)                                     KE
•570 PRINT"[CLEAR][BLUE][12"[DOWN]"]"TAB(
 11)"SPEED [10"."]"S$                          LJ
•580 PRINTTAB(11)"[DOWN]ERRORS[10"."]"E$       AD
•590 PRINTTAB(11)"[DOWN]SCORE[9"."]"SR$        BM
•600 PRINTTAB(11)"[DOWN]HIGH SCORE[4"."]"
 HS$                                           MP
•610 PRINTTAB(8)"[4"[DOWN]"][RED][9"[c P]
 "]"SPC(7)"[8"[c P]"]"                         NL
•620 PRINTTAB(8)"[RVSON]  SPACE  "SPC(7)"
  RETURN "                                     KO
•630 PRINTTAB(8)"[DOWN][BLACK]CONTINUE"SP
 C(10)"END"                                    EH
•700 POKE198,0                                 KB
•710 GETA$:IFA$=""THEN710                      JP
•720 IFA$<>CHR$(13)ANDA$<>CHR$(32)THEN710      GL
•725 POKE53269,0                               PL
•730 IFA$=CHR$(32)THEN500                      GD
•740 POKE792,71:END                            KD
•1000 DATA0,127,128,1,249,192,3,248            HB
•1001 DATA224,3,255,231,3,255,253,35           NC
•1002 DATA255,255,33,255,192,64,243,254        ML
•1003 DATA64,112,0,32,120,0,33,252             MM
•1004 DATA0,71,252,0,79,254,0,111              PB
•1005 DATA246,0,111,243,128,63,224,64          HI
•1006 DATA31,192,0,7,224,0,30,96               GL
•1007 DATA0,48,48,0,60,30,0,21                 IB
•1008 DATA1,255,0,3,243,128,7,241              PM
•1009 DATA192,7,255,206,7,255,250,7            FB
•1010 DATA255,254,99,255,128,17,243,252        IB
•1011 DATA16,112,0,32,120,0,65,248             FD
•1012 DATA0,135,248,0,143,252,0,79             MB
•1013 DATA252,0,111,246,0,63,226,0             EO
•1014 DATA31,194,0,7,192,0,3,128               PJ
•1015 DATA0,7,0,0,15,224,0,21                  PG
•1016 DATA0,127,0,1,249,199,3,248              HM
•1017 DATA253,3,255,255,3,255,240,3            MM
•1018 DATA255,224,17,255,192,16,247,128        PA
•1019 DATA32,115,128,32,113,128,65,248         CO
•1020 DATA192,71,248,96,79,252,0,111           KL
•1021 DATA246,0,111,243,128,63,224,64          HI
•1022 DATA31,192,0,7,224,0,30,96               GL
•1023 DATA0,48,048,0,60,30,0                   NH
•2000 DATAKNIGHT,MAIDEN,KING,QUEEN,PRINCE
 ,SWORD,ZEBRA,VISE,PEASANT,KNAVE               LA
•2005 DATALANCE,BUCKET,QUILT,FLAGON,WIZAR
 D,BIRD,HUT,BRIDGE,PIKE,MAN                    AJ
•2010 DATAHEX,CHILD,RIVER,CHEST,GOLD,SILV
 ER,COIN,CASTLE,TUNIC,SPEAR                    OP
•2015 DATAPATH,MAIL,JOKER,HELMET,FLASK,BE
 AST,ANIMAL,MAZE,BAY,LORD                      OC
•2020 DATATORCH,TABLE,CHAIR,STABLE,HORSE,
 BISHOP,JESTER,CROWN,GAME,BOOK                 BG
•2025 DATAYEOMAN,XEBEC,GORGE,PIT,SHIELD,A
 RMOUR,THIEF,YAM,COOK,SHOE                     FD
•2030 DATALIVER,MALLET,CART,TAX,RAVINE,QU
 AIL,BOW,ARROW,HEN,FIRE                        IA
•2035 DATABABY,WINE,JEWEL,DUNGEON,KNIFE,S
 ERF,CROSS,DAMSEL,QUIVER,HAZE                  CP
•2040 DATAVIPER,LION,DOG,CANDLE,WOLF,SHIP
 ,LOG,GRATE,NUT,CANNON                         DN
•2045 DATAQUARRY,IBEX,KEY,KID,BAZAAR,CLOA
 K,GEM,LANTERN,STAG,SCARF                      NH
```

AUTO-GEN
FROM PAGE 59

```
•1000 PRINTSPC(15)"AUTO-GEN"                   EN
•1010 PRINTSPC(13)"SEP[3" "]1, 1984"           NE
•1020 PRINTSPC(13)"DAVID A JONES"              JF
•1030 REM CREATES DISK FILE OF DATA STATE
MENTS CONTAINING IMAGE OF MEMORY               BM
•1040 :                                        DI
•1050 FL=0        : REM FIRST MEMORY LOCA
TION TO BE SAVED                               GG
•1060 LL=0        : REM LAST MEMORY LOCAT
ION TO BE SAVED                                BC
•1070 LN=1000     : REM OUTPUT PROGRAM LI
NE NUMBER                                      MB
•1080 LB=0        : REM LOW BYTE OF 16 BI
T NUMBER                                       BH
•1090 HB=0        : REM HIGH BYTE OF 16 B
IT NUMBER                                      BO
•1100 HH=0        : REM TEMP STORE FOR HE
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
XIDECIAML CONVERSION                        PD
•1110 MM=0         : REM DECIMAL VALUE OF
NUMBER CONVERTED FROM STRING                MA
•1120 ND=10        : REM NUMBER OF DATA IT
EMS PER LINE                                HC
•1130 NL=2049      : REM POINTER TO START
OF NEXT BASIC LINE                          NG
•1140 X=0          : REM OUTER LOOP COUNTE
R                                           EE
•1150 Y=0          : REM INNER LOOP COUNTE
R                                           JE
•1160 Z=0          : REM DATA ITEM COUNTER  MB
•1170 C$=","       : REM COMMA              LH
•1180 D$=CHR$(131): REM BASIC TOKEN FOR "
DATA"                                       MG
•1190 S$=" "       : REM SPACE              ML
•1200 FI$=""       : REM OUTPUT FILE NAME   KI
•1210 NN$=""       : REM NUMBER STRING INP
UT                                          MP
•1220 X$=""        : REM 4 CHARACTER STRIN
G REPRESENTING 1 MEMORY LOCATION            AO
•1230 :                                     DI
•1240 :      REM MAIN PROGRAM               AK
•1250 :                                     DI
•1260 PRINT:INPUT"FIRST MEMORY LOCATION";
NN$                                         CH
•1270 MM=VAL(NN$)                           CC
•1280 IF LEFT$(NN$,1)="$" THEN GOSUB 1830   AI
•1290 FL=MM                                 NM
•1300 PRINT:INPUT"LAST MEMORY LOCATION";N
N$                                          IN
•1310 MM=VAL(NN$)                           CC
•1320 IF LEFT$(NN$,1)="$" THEN GOSUB 1830   AI
•1330 LL=MM                                 OO
•1340 PRINT:INPUT"STARTING LINE NUMBER";L
N                                           AH
•1350 PRINT:INPUT"NUMBER OF DATA ITEMS PE
R LINE";ND                                  EN
•1360 PRINT:INPUT"OUTPUT FILENAME";FI$      CM
•1370 :                                     DI
•1380 OPEN 15,8,15                          AM
•1390 OPEN 8,8,8,"0:"+FI$+",P,W"            NJ
•1400 GOSUB 1920                            GF
•1410 :                                     DI
•1420 PRINT#8,CHR$(1);CHR$(8);: REM PROGR
AM LOAD LOCATION                            AD
•1430 :                                     DI
•1440 :      REM MAIN LOOP                  BI
•1450 :                                     DI
•1460 FOR X=FL TO LL STEP ND                KL
•1470 NL=NL+2+2+1+4*ND+1                    JA
•1480 LB=NL-256*(INT(NL/256)): HB=INT(NL/
256)                                        KL
•1490 PRINT#8,CHR$(LB);CHR$(HB);: REM NEX
T BASIC LINE                                KG
•1500 LB=LN-256*(INT(LN/256)): HB=INT(LN/
256)                                        OP
•1510 PRINT#8,CHR$(LB);CHR$(HB);: REM NEX
T LINE NUMBER                               DE
•1520 PRINT LN;: REM SHOW USER WHAT'S HAP
PENING                                      FL
•1530 PRINT#8,D$;: REM DATA TOKEN           BL
•1540 :                                     DI
•1550 FOR Y=X TO X+ND-1: REM 0 TO 9 = 10
ITEMS                                       MB
•1560 X$=RIGHT$("[4" "]"+STR$(PEEK(Y)),3)   NM
•1570 :                                     DI
•1580 FOR I=1 TO 3                          IM
•1590 PRINT#8,CHR$(ASC(MID$(X$,I,1)));      OE
•1600 NEXT I                                MN
•1610 :                                     DI
•1620 Z=Z+1                                 EB
•1630 IF Z<ND THEN PRINT#8,C$;              IA
•1640 NEXT Y                                NN
•1650 :                                     DI
•1660 PRINT#8,CHR$(0);: REM END OF LINE T
ERMINATOR                                   AP
•1670 LN=LN+10                              OK
•1680 Z=0                                   EH
•1690 GOSUB 1920                            GF
•1700 NEXT X                                NK
•1710 :                                     DI
•1720 :      REM ADD END OF PROGRAM TERMIN
ATOR                                        JM
•1730 :                                     DI
•1740 PRINT#8,CHR$(0);CHR$(0);              DK
•1750 CLOSE 8                               NI
•1760 GOSUB 1920                            GF
•1770 CLOSE 15                              AB
•1780 PRINT:PRINT"COMPLETE"                 BJ
•1790 END                                   IC
•1800 :                                     DI
•1810 :      REM HEX TO DECIMAL             EM
•1820 :                                     DI
•1830 MM=0: HH=0                            DM
•1840 FOR X=2 TO LEN(NN$)                   CI
•1850 HH=ASC(MID$(NN$,(X),1))-48            IP
•1860 MM=16*MM+HH+7*(HH>9)                  JO
•1870 NEXT                                  IA
•1880 RETURN                                IM
•1890 :                                     DI
•1900 :      REM DISK I/O ERROR             DG
•1910 :                                     DI
•1920 INPUT#15,EN$,EM$,ET$,ES$: EN=VAL(EN
$)                                          IM
•1930 IF EN=0 THEN 1970                     IO
•1940 PRINT"DISK I/O ERROR ";               OM
•1950 PRINT EN$,EM$,ET$,ES$                 PG
•1960 CLOSE 8: CLOSE 15: STOP               NN
•1970 RETURN                                IM
```

MOXEY'S PORCH
FROM PAGE 73

```
·10 POKE52,48:POKE56,48:CLR                     HJ
·20 REM * MOXEYS PORCH * BY BOB BLACKMER        NK
·30 DIMMV(136):HS=0:A=2:B=39:C=41:MP$="MO
XEYS PORCH":PS=53270                            PO
·40 V=53248:S=54272:M=56320:POKEV+21,65:P
OKE53280,6:POKE53281,11:U=1401                  MN
·50 FORL=STOS+24:POKEL,0:NEXTL                  HA
·60 POKES+5,21:POKES+1,20:POKES+6,240:POK
ES+4,17:GOSUB1110                               AJ
·70 PRINT"[CLEAR][HOME][DOWN][DOWN][RVSON
][YELLOW]DO YOU WANT INSTRUCTIONS? (Y/N)
"                                               DH
·80 GETI$:IFI$=""THEN80                         HD
·90 IFI$="N"THEN300                             IH
·100 IFI$<>"Y"THEN80                            AD
·110 PRINT"[CLEAR][HOME]"TAB(11)"[c 6]"MP
$"[DOWN][DOWN]"                                 KN
·120 PRINT"[DOWN][CYAN]ENZO[7" "]JUST GOT
 A POGO STICK AND"                              GH
·130 PRINTTAB(11)"JUMPED OVER TO HIS NEIG
HBORS"                                          KP
·140 PRINTTAB(11)"MULTI-STEP FRONT PORCH.
[3"[DOWN]"]"                                    AI
·150 PRINT"[YELLOW]MOXIE[6" "]DOESN'T LIK
E THE MARKS"                                    KA
·160 PRINTTAB(11)"ENZO'S POGO STICK IS MA
KING"                                           JL
·170 PRINTTAB(11)"ON HIS PORCH.[3"[DOWN]"
]"                                              FN
·180 PRINTTAB(3)"[c 8]TO PLAY USE JOYSTIC
K #2 AND MARK UP[5" "]ALL THE STEPS ON";        IE
·190 PRINT" MOXIE'S PORCH,":PRINTTAB(3)"W
HILE TRYING TO AVOID MOXIE'S PURSUIT"           CG
·200 PRINTTAB(3)"[UP]IF SUCCESSFUL YOU AR
E REWARDED WITH"                                BM
·210 PRINTTAB(3)"ANOTHER PORCH AND A FAST
ER MOXIE.[DOWN][DOWN]"                          KJ
·220 POKEV+23,65:POKEV+29,65:POKE2040,P:P
OKE2046,Q:POKEV+37,6:POKEV+38,5                 CL
·230 POKEV,60:POKEV+1,70                        BG
·240 POKEV+12,60:POKEV+13,110                   FP
·250 PRINTTAB(10)"[RVSON][c 6]HIT ANY KEY
 TO BEGIN":FORK=1TO99:NEXTK                     ND
·260 GETS$:IFS$=""THEN280                       HF
·270 IFS$<>""THEN290                            OP
·280 PRINTTAB(10)"[UP]HIT ANY KEY TO BEGI
N[UP]":FORK=1TO99:NEXTK:GOTO250                 EG
·290 POKEV+23,0:POKEV+29,0                      CH
·300 SL=0                                       IA
·310 PRINT"[CLEAR][HOME][3"[DOWN]"]"TAB(7
)"LEVEL (1-9)"                                  GH
·320 PRINT"[DOWN]"TAB(7)"1 IS THE HARDEST
"                                               EK
·330 GETLV$:IFLV$=""THEN330                     DO
·340 IFLV$<"1"ORLV$>"9"THEN330                  FD
·350 LV=VAL(LV$)+1                              CI
·360 PRINT"[CLEAR]":POKE53272,(PEEK(53272
)AND240)+14                                     HB
·370 POKE53282,2:POKE53283,7:POKE53270,PE
EK(53270)OR16                                   HM
·380 X2=203:Y2=166:X1=157:Y1=102:U=1401         JD
·390 IFLV-1<1THENLV=2                           CP
·400 LV=LV-1                                    MF
·410 PRINT"[HOME]"TAB(12)"[3"[DOWN]"][CYA
N]"MP$                                          AA
·420 GOTO450                                    CN
·430 JV=PEEK(M):FR=JVAND16                      ON
·440 JV=15-(JVAND15):RETURN                     CG
·450 PRINT"[HOME][7"[DOWN]"][c 6]"              AH
·460 PRINTTAB(17)"#$"                           GM
·470 PRINTTAB(16)"#%&$"                         KL
·480 PRINTTAB(15)"#%&%&$"                       OO
·490 PRINTTAB(14)"#%&%&%&$"                     DJ
·500 PRINTTAB(13)"#%&%&%&%&$"                   HI
·510 PRINTTAB(12)"#%&%&%&%&%&$"                 BH
·520 PRINTTAB(11)"#%&%&%&%&%&%&$"               FC
·530 PRINTTAB(10)"#%&%&%&%&%&%&%&$"             GN
·540 PRINTTAB(9)"#%&%&%&%&%&%&%&%&$"            HK
·550 PRINTTAB(8)"#%&%&%&%&%&%&%&%&%&$"          HJ
·560 PRINTTAB(8)"'('('('('('('('('('(":PO
KES+24,15                                       BM
·570 FORL=1TO136 STEP2                          MJ
·580 POKES+4,17:POKES+1,55:POKES+4,16           LJ
·590 X2=X2+MV(L):Y2=Y2+MV(L+1)                  NB
·600 FORG=LVTO1STEP-1                           HE
·610 K=1:GOSUB430                               JF
·620 POKE2040,P:IFK<>LVTHENK=K+1:GOTO620        BI
·630 POKE2046,Q                                 ON
·640 REM ***NOTE*** LINES 650-660 MUST BE
 ENTERED USING ABBREVIATIONS                    NG
·650 POKEV,X1:POKEV+1,Y1:IFPEEK(U)=37THEN
POKEU,C:POKEU+1,C+1:SC=SC+1:POKES+4,129:
POKES+4,128                                     OM
·660 POKEV,X1:POKEV+1,Y1:IFPEEK(U)=39THEN
POKEU,43:POKEU+1,44:SC=SC+1:POKES+4,129:
POKES+4,128                                     HG
·670 POKEV+12,X2:POKEV+13,Y2                    KG
·680 P=P+1:IFP>15THENP=13                       IL
·690 Q=Q+1:IFQ>199THENQ=198                     OP
·700 IFP=14THENPOKES+4,129:POKES+4,128          JD
·710 IFJV=4THENX1=X1-16:U=U-A:IFPEEK(U)=3
2THENU=U+A:X1=X1+16                             PO
·720 IFJV=8THENX1=X1+16:U=U+A:IFPEEK(U)=3
2ORPEEK(U)=36THENU=U-A:X1=X1-16                 BP
·730 IFJV=5THENX1=X1-8:Y1=Y1-8:U=U-C:IFPE
EK(U)=32THENU=U+C:X1=X1+8:Y1=Y1+8               EN
·740 IFJV=6THENX1=X1-8:Y1=Y1+8:U=U+B:IFPE
EK(U)=32THENU=U-B:X1=X1+8:Y1=Y1-8               JO
·750 REM ***NOTE*** LINES 760-770 MUST BE
 ENTERED USING ABBREVIATIONS                    DO
·760 IFJV=9THENX1=X1+8:Y1=Y1-8:U=U-B:IFPE
EK(U)=32ORPEEK(U)=36THENU=U+B:X1=X1-8:Y1
=Y1+8                                           PM
```

```
•770 IFJV=10THENX1=X1+8:Y1=Y1+8:U=U+C:IFP
EEK(U)=40ORPEEK(U)=32THENU=U-C:X1=X1-8:Y
1=Y1-8                                      OC
•780 IFSC=55THEN970                         JM
•790 IFY1=Y2ANDX1-2=X2THEN810               AN
•800 NEXTG:NEXTL:GOTO570                    CA
•810 POKEV+23,64:POKEV+29,64:POKES+4,17:P
OKES+6,240:X2=180:Y2=166:POKE2046,199       MJ
•820 FORL=1TO99STEP2                        KJ
•830 X2=X2+MV(L):Y2=Y2+MV(L+1)              NB
•840 POKEV+12,X2:POKEV+13,Y2                KG
•850 POKES+1,L+20                           EL
•860 FORK=1TO40:NEXTK                       MC
•870 NEXTL                                  MO
•880 POKES+24,0:POKEV+23,0:POKEV+29,0:PRI
NT"[CLEAR]":POKEPS+2,21:POKEPS,PEEK(PS)A
ND239                                       BD
•890 SL=SL+SC*10:SC=0:PRINTSPC(12)"[YELLO
W][RVSON]"MP$:PRINTTAB(11)"[DOWN][c 6]YO
UR SCORE:[YELLOW]";SL                       JI
•900 IFSL>HSTHENHS=SL                       BI
•910 PRINTTAB(11)"[DOWN][c 8]HIGH SCORE:"
;HS                                         JM
•920 PRINT"[YELLOW][RVSON][DOWN][DOWN][RI
GHT][RIGHT]WOULD YOU LIKE TO PLAY AGAIN?
 (Y/N)"                                     GD
•930 GETF$:IFF$=""THEN930                   JF
•940 IFF$="N"THENPOKEV+21,0:POKES+24,0:PR
INT"[CLEAR]":END                            FC
•950 IFF$<>"Y"THEN930                       CC
•960 GOTO300                                BP
•970 POKEV+23,1:POKEV+29,1:POKES+4,17:POK
ES+6,240:X1=170:Y1=166                      OI
•980 FORL=1TO99STEP2                        KJ
•990 X1=X1+MV(L):Y1=Y1+MV(L+1)              HF
•1000 POKEV,X1:POKEV+1,Y1                   GE
•1010 POKES+1,L+20                          EL
•1020 FORK=1TO40:NEXTK                      MC
•1030 NEXTL                                 MO
•1040 SL=SL+SC*10                           FM
•1050 POKES+24,0:POKEV+23,0:POKEV+29,0:PR
INT"[CLEAR]":POKEPS+2,21:POKEPS,PEEK(PS)
AND239                                      BD
•1060 PRINTTAB(12)"[DOWN][DOWN][CYAN][RVS
ON]"MP$:PRINTTAB(13)"[DOWN][YELLOW]SCORE
:";SL:SC=0                                  BE
•1070 IFLV-1<1THENLV=2                      CP
•1080 PRINTTAB(10)"[YELLOW][DOWN][DOWN]YO
U HAVE REACHED":PRINTTAB(14)"[DOWN]LEVEL
:[c 6]";LV-1                                CG
•1090 FORK=1TO1500:NEXTK                    AP
•1100 POKEPS,PEEK(PS)OR16:POKEPS+2,(PEEK(
PS+2)AND240)+14:GOTO360                     LH
•1110 PRINT"[CLEAR][HOME][5"[DOWN]"][YELL
OW]I'M WORKING[4"."]PLEASE BE PATIENT"      MO
•1120 POKE56334,PEEK(56334)AND254:POKE1,P
EEK(1)AND251                                BE
•1130 FORI=0TO57                            MF
•1140 FORJ=0TO7                             IC
•1150 POKE14336+I*8+J,PEEK(V+I*8+J)         PM
•1160 NEXTJ:NEXTI                           NB
•1170 POKE1,PEEK(1)OR4:POKE56334,PEEK(563
34)OR1                                      IE
•1180 CK=0:FORCH=35TO45                     PB
•1190 FORBY=0TO7                            NE
•1200 READ NU:CK=CK+NU                      IC
•1210 POKE14336+(8*CH)+BY,NU                JH
•1220 NEXT BY:NEXT CH                       GL
•1230 IFCK<>10450THENPRINT"ERROR IN DATA-
LINES 1300-1400":END                        DP
•1240 CK=0:FORS1=832TO894:READQ1:CK=CK+Q1
:POKES1,Q1:NEXT                             NF
•1250 FORS2=896TO958:READQ2:CK=CK+Q2:POKE
S2,Q2:NEXT                                  HB
•1260 FORS3=960TO1022:READQ3:CK=CK+Q3:POK
ES3,Q3:NEXT                                 JN
•1270 FORS4=12672TO12734:READQ4:CK=CK+Q4:
POKES4,Q4:NEXT                              LJ
•1280 FORS5=12736TO12798:READQ5:CK=CK+Q5:
POKES5,Q5:NEXT:P=13:Q=198                   OP
•1290 IFCK<>8699THENPRINT"ERROR IN DATA-L
INES 1410-1800":END                         MK
•1300 CK=0:FORW=1TO136STEP2:READQ6,Q7:MV(
W)=Q6:MV(W+1)=Q7:CK=CK+Q6+Q7:NEXTW          DA
•1310 IFCK<>0THENPRINT"ERROR IN DATA LINE
S 1810-1870":END                            OG
•1320 POKEV+28,65:POKEV+37,6:POKEV+38,5:P
OKEV+45,0:POKEV+39,15:RETURN                PE
•1330 DATA0,0,0,0,2,10,42,170               CJ
•1340 DATA0,0,0,0,128,160,168,170           MH
•1350 DATA106,90,86,85,85,149,165,169       KO
•1360 DATA171,175,191,255,255,254,250,234   PI
•1370 DATA106,90,86,85,85,21,5,1            KL
•1380 DATA171,175,191,255,255,252,240,192   DA
•1390 DATA104,90,86,85,85,149,165,169       PE
•1400 DATA43,175,191,255,255,254,250,234    HM
•1410 DATA104,90,86,85,85,21,5,1            FB
•1420 DATA43,175,191,255,255,252,240,192    AO
•1430 DATA0,0,0,0,0,0,0,0                   FG
•1440 DATA 0,0,0,0,0,0,0,0                  FG
•1450 DATA 0,0,0,0,0,112,0,0                JP
•1460 DATA 136,0,1,4,0,0,112,0              ND
•1470 DATA 1,84,0,4,81,0,4,169              EA
•1480 DATA 0,1,36,0,0,32,0,1                KP
•1490 DATA 36,0,1,36,0,1,36,0               NN
•1500 DATA 1,36,0,1,36,0,0,168              HF
•1510 DATA 0,0,32,0,0,32,0                  EK
•1520 DATA 0,0,0,0,0,0,0,0                  FG
•1530 DATA 0,0,0,0,0,0,0,0                  FG
•1540 DATA 0,0,0,112,0,0,136,0              GI
•1550 DATA 1,4,0,0,112,0,1,84               PO
•1560 DATA 0,4,81,0,4,169,0,1               NK
•1570 DATA 36,0,0,32,0,1,36,0               NC
•1580 DATA 1,36,0,1,36,0,1,36               CK
•1590 DATA 0,1,36,0,0,168,0                 IA
•1600 DATA 0,0,0,0,0,0,0,112                JP
```

```
•1610 DATA 0,0,136,0,1,4,0,0                  OK
•1620 DATA 112,0,1,84,0,4,81,0                GC
•1630 DATA 4,169,0,1,36,0,0,32                FF
•1640 DATA 0,1,36,0,1,36,0,1                  JG
•1650 DATA 36,0,1,36,0,1,36,0                 NN
•1660 DATA 0,168,0,0,32,0,0,32                AH
•1670 DATA 0,0,0,0,0,0,0,0                    PD
•1680 DATA0,0,0,0,0,0,0,0                     FG
•1690 DATA0,0,0,0,0,0,0,0                     FG
•1700 DATA0,0,0,0,0,0,0,0                     FG
•1710 DATA0,0,0,0,42,0,0,170                  LO
•1720 DATA128,2,235,160,2,170,160,2           PG
•1730 DATA255,224,2,170,160,0,170,128         HC
•1740 DATA0,42,0,0,34,0,2,162                 BC
•1750 DATA160,0,0,0,0,0,0,0                   GB
•1760 DATA0,0,0,0,0,0,0,0                     FG
•1770 DATA0,0,0,0,0,0,0,0                     FG
•1780 DATA0,0,0,0,0,0,0,0                     FG
•1790 DATA0,0,0,0,0,0,0,0                     FG
•1800 DATA0,0,42,0,0,170,128,2                AC
•1810 DATA235,160,2,170,160,2,255,224         JA
•1820 DATA2,191,160,0,170,128,0,42            CK
•1830 DATA0,0,34,0,2,162,160                  LJ
•1840 DATA-16,0,-16,0,-16,0,-16,0,-16,0,8
 ,-8,8,-8,8,-8,8,-8,-8,-8                     GK
•1850 DATA-8,-8,8,-8,8,-8,8,8,8,8,8,8,8,8
 ,-8,8,-8,8,-16,0                             JO
•1860 DATA-16,0,-16,0,-8,8,16,0,16,0,-8,-
 8,-8,-8,8,-8,8,-8,8,-8                       IL
•1870 DATA8,8,8,8,8,8,8,8,-8,8,-8,8,-8,8,
 16,0,16,0,16,0                               PD
•1880 DATA16,0,-8,-8,-8,-8,-16,0,8,8,-8,8
 ,-16,0,-16,0                                 PO
•1890 DATA-16,0,8,-8,8,-8,-8,-8,8,-8,-16,
 0,-16,0,-16,0,-8,8                           GF
•1900 DATA-8,8,-8,8,-8,8,16,0,16,0,16,0,1
 6,0,8,-8,16,0,16,0,16,0                      IJ
```

FILE LOCK

FROM PAGE 62

```
•5 REM FILE LOCK V0222/85                     KI
•10 GOTO 10000                                IE
•50 POKE 198,0                                KB
•52 GET A$:IF A$="" THEN GOTO 52              NH
•55 RETURN                                    IM
•70 REM WAIT FOR RETURN                       JL
•80 POKE 198,0:POKE 204,0                     HG
•85 GET A$:POKE 207,0:IF A$<>CHR$(13) THE
N GOTO 85                                     DB
•90 POKE 204,1:PRINT " "CHR$(157);:RETURN     KP
•100 REM WAIT FOR KEY                         JJ
•105 POKE 198,0:POKE 204,0                    HG
•110 GET A$:POKE 207,0:IF A$="" THEN GOTO
 110                                          CM
•115 POKE 204,1:PRINT " "CHR$(157);:RETUR
N                                             KP
•150 REM BOX ROUTINE                          JJ
•152 GOSUB 190                                CO
•155 PRINT CHR$(117);:FOR I=0 TO W-2:PRIN
T CHR$(99);:NEXT I:PRINT CHR$(105);           MC
•158 FOR J=0 TO H-2:RW=RW+1:GOSUB 190         LI
•160 PRINT CHR$(103);:FOR I=0 TO W-2:PRIN
T " ";:NEXT I:PRINT CHR$(104);                FG
•161 NEXT J                                   MM
•162 RW=RW+1:GOSUB 190                        EF
•165 PRINT CHR$(106);:FOR I=0 TO W-2:PRIN
T CHR$(99);:NEXT I:PRINT CHR$(107);           PE
•170 PRINT BL$;:RETURN                        EF
•190 REM SET CURSOR LOCATIOM                  OH
•195 POKE 214,RW:POKE 211,CL:SYS 58640:RE
TURN                                          AF
•200 REM READ DIRECTORY                       HI
•210 OPEN 15,8,15,"I0:":OPEN 2,8,2,"#"        EI
•220 TK=18:SC=1:NF=0                          DO
•230 PRINT#15,"U1";2;0;TK;SC                  LN
•240 GET#2,A$:A$=A$+CHR$(0):T=ASC(A$)         NL
•250 GET#2,A$:A$=A$+CHR$(0):S=ASC(A$)         AG
•260 FOR F=0 TO 7:REM 8 ENTRIES/SEC           LM
•270 PRINT#15,"B-P";2;F*32+2                  NK
•275 GET#2,A$:IF A$="" THEN A$=CHR$(0)        LB
•280 A=ASC(A$):IF ASC(A$)=0 THEN GOTO 325     BK
•285 L$=">":IF (A AND 64) =0 THEN L$="[SS
]"                                            OA
•290 GET#2,A$:GET#2,A$                        GE
•300 F$=L$+""                                 ID
•305 FOR I=0 TO  15:GET#2,A$:IF A$="" THE
N A$=CHR$(0)                                  CD
•310 IF ASC(A$)=160 THEN GOTO 320             BM
•315 F$=F$+A$                                 IO
•320 NEXT I                                   MN
•322 FL$(NF)=F$:S(NF)=SC:EN(NF)=F:NF=NF+1     IP
•325 NEXT F                                   MI
•330 IF T<>0 THEN TK=T:SC=S:GOTO 230          CJ
•335 CLOSE2:CLOSE15                           BI
•340 NF=NF-1:RETURN                           IA
•400 REM PRINT FORMATTED SCREEN               KA
•405 PRINT CHR$(147);:PRINT HD$:PRINT         KD
•410 RW=2:CL=0:H=20:W=19:GOSUB 150:           BM
•415 RW=2:CL=20:H=20:W=19:GOSUB 150           GC
•450 REM FILL SCREEN WITH FILES               EC
•455 RW=3:CL=1:GOSUB 190:ZN=0                 GE
•460 FOR J=0 TO 18:I=PG*38+ZN*19+J            CM
•462 IF SF(I)=0 THEN PRINT FL$(I);:GOTO 4
64                                            OG
•463 IF SF(I)=1 THEN PRINT RV$FL$(I)RO$;      FM
•464 RW=RW+1:GOSUB 190:NEXT J                 HF
•465 RW=3:CL=21:GOSUB 190:ZN=1                MD
•470 FOR J=0 TO 18:I=PG*38+ZN*19+J            CM
•472 IF SF(I)=0 THEN PRINT FL$(I);:GOTO 4
74                                            PD
```

```
•473 IF SF(I)=1 THEN PRINT RV$FL$(I)RO$;  FM
•474 RW=RW+1:GOSUB 190:NEXT J             HF
•475 ZN=0:RETURN                          EO
•500 REM SCRATCH                          KM
•510 NC$="SCRATCH  FILES "                KJ
•515 GOSUB 800                            CI
•519 OPEN 15,8,15                         AM
•520 FOR I=0 TO NF:IF SF(I)=0 THEN GOTO 5
 40                                       JB
•525 F$=FL$(I):IF LEFT$(F$,1)=">" THEN GO
 TO 540                                   KP
•530 F$=RIGHT$(F$,LEN(F$)-1)              BL
•535 PRINT#15,"S0:"F$:GOSUB 900           NM
•540 NEXT I                               MN
•550 CLOSE 15                             AB
•599 RETURN                               IM
•600 REM LOCK                             MA
•610 NC$="[3" "]LOCK  FILES "             EB
•615 GOSUB 800                            CI
•619 OPEN 15,8,15:OPEN 2,8,2,"#"          EI
•620 FOR I=0 TO NF:IF SF(I)=0 THEN GOTO 6
 60                                       JI
•630 GOSUB 690                            CP
•635 GET#2,A$:IF A$="" THEN A$=CHR$(0)    LB
•640 FT=ASC(A$):FT=(FT OR 64)             GK
•650 GOSUB 680                            CO
•655 GOSUB 900                            DB
•660 NEXT I                               MN
•670 CLOSE 2:CLOSE15:RETURN               ON
•680 GOSUB 695:PRINT#2,CHR$(FT);          DO
•684 PRINT#15,"B-P";2;0                   AF
•685 PRINT#15,"U2";2;0;18;S(I):RETURN     IJ
•690 PRINT#15,"U1";2;0;18;S(I)            LP
•695 PRINT#15,"B-P";2;EN(I)*32+2:RETURN   NM
•699 RETURN                               IM
•700 REM UNLOCK                           GE
•705 RW=1:CL=12:GOSUB 190                 PA
```



```
•710 NC$=" UNLOCK  FILES "                KA
•715 GOSUB 800                            CI
•719 OPEN 15,8,15:OPEN 2,8,2,"#"          EI
•720 FOR I=0 TO NF:IF SF(I)=0 THEN GOTO 6
 60                                       JI
•730 GOSUB 690                            CP
•735 GET#2,A$:IF A$="" THEN A$=CHR$(0)    LB
•740 FT=ASC(A$):FT=(FT AND 191)           JA
•750 GOSUB 680:GOSUB 900                  KD
•760 NEXT I                               MN
•770 CLOSE 2:CLOSE15:RETURN               ON
•800 REM SELECT FILES                     LL
•805 RW=1:CL=12:GOSUB 190:PRINT YL$NC$;   HI
•806 RW=24:CL=0:GOSUB 190:PRINT CD$;      KM
•807 RW=3:CL=18:GOSUB 190:PRINT AR$;:J=0  CE
•810 GOSUB 50                             PP
•811 IF A$=CM$ THEN RETURN                IL
•812 IF A$=CHR$(13) THEN GOSUB   820:GOTO
 810                                      EF
•813 IF A$=AR$ THEN GOTO 10000:REM EXIT A
 ND START OVER                            KN
•814 IF A$=CHR$(19)  THEN GOSUB 190:PRINT
 " ";:GOSUB 1000:GOTO 805                 MO
•815 IF A$=CHR$(157) THEN GOSUB  890:GOTO
 810                                      GF
•816 IF A$=CHR$(17)  THEN GOSUB  840:GOTO
 810                                      DL
•817 IF A$=CHR$(145) THEN GOSUB  860:GOTO
 810                                      GF
•818 IF A$=CHR$(29)  THEN GOSUB  880:GOTO
 810                                      GM
•819 GOTO 810                             CN
•820 TP=CL:CL=CL-17:I=PG*38+ZN*19+J       MP
•821 IFSF(I)=1THENGOSUB190:PRINTCHR$(146)
 FL$(CF+I);:SF(I)=0:GOTO 824              EA
•823 IF SF(I)=0 THEN GOSUB 190:PRINT CHR$
 (18)FL$(CF+I)CHR$(146);:SF(I)=1          OO
•824 CL=TP:RETURN                         LM
•840 I=PG*38+ZN*19+J:IF I=NF THEN RETURN  KD
•841 J=J+1:IF J>19 THEN J=19              KJ
•844 GOSUB 190:PRINT " ";                 JI
•845 RW=RW+1:IF RW>21 THEN RW=21:J=J-1    EK
•850 GOSUB 190:PRINT AR$;:RETURN          KL
•860 J=J-1:IF J<0 THEN J=0                PD
•861 GOSUB 190:PRINT " ";                 JI
•865 RW=RW-1:IF RW<3 THEN RW=3            EF
•870 GOSUB 190:PRINT AR$;:RETURN          KL
•880 IF NF<PG*38+19  THEN RETURN          LF
•885 GOSUB 190:PRINT " ";:RW=3:CL=38:J=0:
 ZN=1:GOSUB 190:PRINT AR$;:RETURN         FK
•890 REM                                  JD
•895 GOSUB 190:PRINT " ";:RW=3:CL=18:J=0:
 ZN=0:GOSUB 190:PRINT AR$;:RETURN         KJ
•900 INPUT#15,A$,B$,C$,D$                 CN
•905 RW=23:CL=0:GOSUB 190:PRINT "[38" "]"
 ;                                        MO
•910 RW=23:CL=0:GOSUB 190:PRINT A$"[SS]"B
 $"[SS]"C$"[SS]"D$;:ER=1:RETURN           MK
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
·920 ER=0:RETURN                                    EN
·1000 IF NF<38 THEN RETURN                          IJ
·1005 PG=PG+1:IF PG*38>NF THEN PG=0                 MF
·1010 GOSUB 400:RETURN                              OJ
·10000 CLR:DIM FL$(76),S(76),EN(76),SF(76
)                                                   EP
·10005 BL$=CHR$(154):YL$=CHR$(158):RV$=CH
R$(18):RO$=CHR$(146):AR$=CHR$(95)                   BE
·10010 POKE 53280,11:POKE 53281,0:PRINT C
HR$(147)BL$;                                        ED
·10012 CD$=BL$+"  <CRSR> <HOME>"+YL$+" MO
VES "+BL$+" <RTRN>"+YL$+" SELECTS [SS]"+
BL$                                                 EH
·10015 HD$=CHR$(18)+"[10" "]FILE LOCK V0[
3"2"]/85[12" "]"+CHR$(146)                          NO
·10020 PRINT HD$;                                   IG
·10030 PRINT:PRINT "INSERT DISK AND HIT "
YL$"<RETURN>"BL$;                                   LC
·10035 GOSUB 70:PRINT:PRINT "READING DIRE
CTORY":                                             MH
·10040 GOSUB 200:PG=0:ZN=0:GOSUB 400                NK
·10050 RW=24:CL=0:GOSUB 190                         PG
·10055 PRINTYL$ "SCRATCH LOCK UNLOCK PAGE
 BOOT QUIT ?  "BL$CHR$(157);                        IE
·10056 GOSUB 100                                    CJ
·10060 IF A$="Q" THEN END                           GE
·10065 IF A$="S" THEN PRINT A$;:CM$=A$:GO
SUB 500:GOTO 10000                                  JL
·10070 IF A$="L" THEN PRINT A$;:CM$=A$:GO
SUB 600:GOTO 10000                                  OJ
·10075 IF A$="U" THEN PRINT A$;:CM$=A$:GO
SUB 700:GOTO 10000                                  GH
·10080 IF A$="B" THEN PRINT A$;:GOTO 1000
0                                                   IH
·10085 IF A$="P" THEN PRINT A$;:GOSUB 100
0                                                   MG
·10090 GOTO 10050                                   IF
·60000 CLOSE 15:OPEN 15,8,15,"I0:":CLOSE1
5                                                   IA
·60001 SAVE "@0:FILE LOCK V0[3"2"]",8               GA
·61000 OPEN 15,8,15:INPUT#15,A$,B$,C$,D$            GK
·61001 PRINT:PRINT A$"[SS]"B$"[SS]"C$"[SS
]"D$:CLOSE15:STOP                                   AC
```

ROCKETS, BOATS & PIGS IN POKES

FROM PAGE 18 — PIG IN A POKE

```
·1 REM    *** PIG IN A POKE ***                     IE
·2 REM AN EXAMPLE OF GAME PROGRAMMING WIT
H A SPRITE                                          OE
·4 REM ** MOVE TOP OF MEMORY TO MAKE ROOM
 FOR VIDEO BLOCK AT 32768                           CA
·5 POKE 55,255:POKE 56,127:POKE 643,255:P
OKE 644,127                                         KG
·9 REM       *** SET-UP SUBROUTINES ***             OF
·10 GOSUB 600:REM (VIDEO MEMORY)                    II
·11 GOSUB 700:REM (BACKGROUND STRING)               BM
·12 GOSUB 800:REM (SPRITE SHAPES)                   PB
·13 GOSUB 900:REM (SPRITE POSITIONS)                IE
·15 GOTO 100                                        CF
·98 REM        *** ACTION LOOP ***                  KJ
·100 K=KP(PEEK(653)):IF K>2 OR EG=1 THEN
300                                                 NO
·105 IF K>1 THEN 130                                AG
·110 HP=HP+K(K):IF HP<0 THEN HP=HZ                  HE
·115 IF HP>HZ THEN HP=0                             KM
·120 IF K<>XK THEN XK=K:POKE LT(0),ST(XK)           KN
·125 POKE HT(0),HH(HP):POKE HR,HB(HP)               AO
·130 IF VAL(TI$)>=DT THEN GOSUB 500                 EB
·135 IF PEEK(CF)>0 THEN GOSUB 400                   NA
·196 GOTO 100                                       CF
·198 REM       *** WAIT FOR RESPONSE ***            HO
·296 RETURN                                         IM
·298 REM       *** ENDING ROUTINES ***              NL
·300 PRINT "[CLEAR]FINAL SCORE:  "RT*100+
VP*10:PRINT:PRINT:VP=15:POKE VT(0),VV(VP
)                                                   NM
·310 PRINT "TO PLAY AGAIN, PRESS P":PRINT
 "TO QUIT, PRESS Q":HP=79                           JE
·315 HP=79:POKE HR,HB(HP):POKE HT(0),HH(H
P):XP=1:POKE LT(0),ST(XP)                           JM
·320 Z=PEEK(203):IF Z=62 THEN 370                   EA
·325 POKE HT(0),HH(HP):POKE HR,HB(HP)               AO
·330 HP=HP+1:IF HP>HZ THEN HP=0                     FJ
·345 IF Z<>41 THEN 320                              PE
·350 EG=0:DL=0:DT=1:DS=5:RT=0:RC=0:GOSUB
930:GOTO 100                                        AK
·369 REM ** PUT VIDEO MEMORY BACK TO FIRS
T BLOCK, AND SCREEN MEMORY TO 1024                  EG
·370 POKE 56578,PEEK(56578)OR3:POKE 56576
,(PEEK(56576)AND 252)OR 3                           MB
·371 POKE 53272,20:POKE 648,4                       OM
·379 REM ** REENABLE SHIFT/COMMODORE AND
RUN-STOP/RESTORE                                    FB
·380 POKE 657,0:POKE 792,71:POKE 808,237            PK
·390 END                                            IC
·398 REM       *** COLLISION HANDLER ***            IG
·399 REM ** END GAME?                               CE
·400 VP=VP-1:IF VP<0 THEN EG=1:RETURN               PN
·401 REM ** MOVE UP                                 NJ
·402 POKE VT(0),VV(VP)                              MD
·403 REM ** CLEAR COLLISION REGISTER                BF
·404 Z=PEEK(CS):Z=PEEK(CF)                          KI
·405 REM ** NEW COLLISION?                          ON
·406 IF PEEK(CF)>0 THEN 400                         FC
·496 RETURN                                         IM
·498 REM       *** SCROLL ROUTINE ***               JF
·499 REM ** NEW ROW OR SIMPLE JUMP?                 PJ
·500 IF RC=DS THEN 506                              NB
```

```
·501 REM ** SIMPLE JUMP                      NJ
·502 PRINT "[HOME][25"[DOWN]"][HOME]";       OJ
·503 RC=RC+1:GOTO 520                        CK
·505 REM ** PUT NEW ROW ON SCREEN            BG
·506 PRINT "[HOME][24"[DOWN]"]"MID$(WS$,W
L(DL)+INT(19*RND(9)),40)"[HOME]";            PL
·508 RC=0:RT=RT+1:DL=DL+1:IF DL>8 THEN DL
=0:DS=DS-1                                   AJ
·509 REM ** END OF GAME?                     LE
·510 IF RT>49 THEN EG=1                      GG
·520 IF PEEK(CF)>0 THEN GOSUB 400:REM **
COLLISION                                    KN
·580 POKE 217,L1:POKE 218,L2                 PJ
·589 REM ** RESET TIMER                      NP
·590 TI$="[6"0"]"                            NI
·596 RETURN                                  IM
·598 REM     *** ARRANGE MEMORY ***          KB
·599 REM USE THIRD VIDEO BLOCK (32768 TO
49151), SO ROM CHARACTER SET IS USABLE.      AL
·600 VB=32768:POKE 56578,PEEK(56578)OR3:P
OKE 56576,(PEEK(56576)AND 252)OR 1           CN
·601 REM ** TELL VIC-2 WHERE SCREEN IS WI
THOUT CHANGING CHARACTER SET LOCATION        GA
·602 SB=0:POKE 53272,(SB*16)+4:SB=VB+1024
*SB                                          MO
·603 REM ** TELL BASIC WHERE SCREEN IS       GB
·604 BB=SB/256:POKE 648,BB                   AK
·608 REM     *** REGISTER ADDRESSES ***      EJ
·609 REM ** SPRITE SHAPE LOCATION TABLE      IF
·610 LT(0)=SB+1016:FOR I=1 TO 7:LT(I)=LT(
I-1)+1:NEXT                                  AF
·611 REM ** SPRITE COLOR TABLE               LA
·612 CT(0)=53287:FOR I=1 TO 7:CT(I)=CT(I-
1)+1:NEXT                                    AN
·613 REM ** SPRITE HORIZONTAL POSITION TA
BLE (LOW BYTES)                              CH
·614 HT(0)=53248:FOR I=1 TO 7:HT(I)=HT(I-
1)+2:NEXT                                    KC
·615 REM ** SPRITE VERTICAL POSITION TABL
E                                            IJ
·616 VT(0)=53249:FOR I=1 TO 7:VT(I)=VT(I-
1)+2:NEXT                                    AJ
·617 REM ** SPRITE HORIZONTAL HIGH-BIT RE
GISTER                                       IG
·618 HR=53264                                FB
·619 REM ** SPRITE ENABLE REGISTER           AF
·620 ES=53269                                FE
·621 REM ** VERTICAL EXPANSION REGISTER (
1=DOUBLE HEIGHT)                             IB
·622 VE=53271                                GC
·623 REM ** HORIZONTAL EXPANSION REGISTER
 (1=DOUBLE WIDTH)                            PL
·624 HE=53277                                FC
·625 REM ** SPRITE PRIORITY REGISTER (1=S
PRITE IS IN FRONT OF FOREGROUND)             GM
·626 PR=53275                                FJ
·627 REM ** MULTICOLOR ENABLE REGISTER (1
=MULTI-COLOR ENABLED)                        IM
·628 EM=53276                                EK
·629 REM ** SPRITE MULTICOLOR COLOR REGIS
TERS                                         OH
·630 MR=53285:REM ('01' REGISTER: ADD 1 T
O MR FOR '11' REGISTER)                      LJ
·631 REM ** SPRITE-SPRITE COLLISION REGIS
TER                                          OF
·632 CS=53278                                GA
·633 REM ** SPRITE-FOREGROUND COLLISION R
EGISTER                                      MD
·634 CF=53279                                FK
·635 REM ** SET-BIT AND CLEAR-BIT VALUES     BI
·636 BS(0)=1:FOR I=1 TO 7:BS(I)=2*BS(I-1)
:NEXT                                        NE
·637 FOR I=0 TO 7:BC(I)=255-BS(I):NEXT       LB
·638 REM     *** INITIALIZE VALUES ***       DK
·639 REM ** FOREGROUND COLOR                 FL
·640 POKE 53281,9:PRINT "[CLEAR][c 2]";:R
EM (BROWN)                                   BC
·641 REM ** BACKGROUND COLOR                 DK
·642 POKE 53281,5 :REM (GREEN)               GM
·643 REM ** BORDER COLOR                     BE
·644 POKE 53280,9:REM (BROWN)                LM
·645 REM ** SPRITE COLORS (DEFAULTS: WHI,
RED,L-GRN,PUR,GRN,BLU,YEL,M-GRAY)            IE
·646 POKE CT(0),1:REM (THE PIG IS WHITE)     FI
·647 REM ** SET PRIORITY                     EF
·648 POKE PR,0:REM (ALL IN FRONT)            OD
·649 REM ** SET HORIZONTAL SIZES             BJ
·650 POKE HE,0:REM (ALL NORMAL WIDTH)        AL
·651 REM ** SET VERTICAL SIZES               CC
·652 POKE VE,0:REM (ALL NORMAL HEIGHT)       PK
·653 REM ** ENABLE SPRITES                   OD
·654 POKE ES,BS(0):REM (SPRITE 0 ONLY)       KB
·689 REM     *** SAFETY PROCEDURES ***       CM
·690 POKE 657,128:REM DISABLE SHIFT/COMMO
DORE CHARACTER SET SWITCH                    OA
·691 POKE 808,234:POKE 792,193:REM DISABL
E STOP AND STOP/RESTORE                      JJ
·696 RETURN                                  IM
·698 REM     *** BACKGROUND STRING ***       FD
·700 WC$="[s V]":BC$=" "                     IH
·709 REM SET UP WALL STRING                  GA
·710 FOR I=1 TO 12                           MP
·714 REM EACH WALL UNIT IS WIDER, EACH GA
P NARROWER                                   CD
·715 FOR J=1 TO 5+I:WS$=WS$+WC$:NEXT         EA
·720 FOR J=1 TO 15-I:WS$=WS$+BC$:NEXT        GK
·725 NEXT                                    IA
·729 REM SET WALL STARTING POINTS            JM
·730 WL(0)=1:FOR I=1 TO 9:WL(I)=WL(I-1)+2
0:NEXT                                       PD
·739 REM SET DIFFICULTY LEVELS (DL=SPACIN
G; DT=TIMING; DS=VERTICAL SPACING)           PI
·740 DL=0:DT=1:DS=5                          OJ
·796 RETURN                                  IM
·798 REM     *** SPRITE SHAPES ***           CD
·799 REM ** NUMBER OF SPRITE SHAPES (-1);
```

```
NUMBER OF LINES PER SPRITE SHAPE (-1)       CE
•800 NS=1:TS=9                              GK
•801 REM ** LOCATE SPRITE SHAPE MEMORY      OM
•802 ST(0)=16:FOR I=1 TO NS:ST(I)=ST(I-1)
 +1:NEXT                                    HM
•803 REM ** LOOPS                           HL
•804 FOR I=0 TO NS:FOR J=0 TO TS:READ SS$
 :FOR K=0 TO 2:X=0:Y=VB+ST(I)*64+J*3+K      IB
•805 REM ** CONVERT STRINGS TO SHAPE BYTE
S (USE BS(0-7) TO SET BITS)                 LA
•806 FOR L=1 TO 8:M=L+8*K:M=ASC(MID$(SS$,
M,1)):IF M=42 THEN X=X OR BS(8-L)           KC
•807 NEXT:POKE Y,X:NEXT:PRINT SS$:NEXT      GO
•808 Z=64-(TS*3+2):FOR J=Y+1 TO Y+Z:POKE
J,0:NEXT:NEXT                               PF
•809 REM SPRITE SHAPE DATA, 21 LINES  PER
 SHAPE: *='ON'  .='OFF'                     MJ
•810 DATA "[5"."]*[14"."]**.."              HG
•811 DATA "[5"."]**[4"."][6"*"][5"."]*."    GO
•812 DATA "[4"."][16"*"].*.."               IG
•813 DATA "[3"."]*.[16"*"][3"."]"           MG
•814 DATA "[22"*"].."                       LK
•815 DATA "[22"*"].."                       LK
•816 DATA "[4"."][17"*"][3"."]"             HO
•817 DATA "[6"."][14"*"][4"."]"             HC
•818 DATA "[6"."]**.*[5"."]*.**[5"."]"      AC
•819 DATA "[5"."]**..**[3"."]**.**[5"."]"   DC
•820 REM ** 2ND SHAPE                       CL
•821 DATA ".**[14"."]*[6"."]"               JG
•822 DATA "*[5"."][6"*"][4"."]**[6"."]"     FO
•823 DATA ".*.[16"*"][5"."]"                MG
•824 DATA "..[16"*"].*[4"."]"               IG
•825 DATA ".[22"*"]."                       BK
•826 DATA ".[22"*"]."                       BK
•827 DATA "..[17"*"][5"."]"                 HO
•828 DATA "[3"."][14"*"][7"."]"             KC
•829 DATA "[4"."]**.*[5"."]*.**[7"."]"      MC
•830 DATA "[4"."]**.**[3"."]**..**[6"."]"   OC
•896 RETURN                                 IM
•898 REM     *** SPRITE POSITIONS ***       KA
•899 REM ** POSSIBLE POSITIONS DIM'ED       MK
•900 VZ=19:HZ=147:DIM HH(HZ),VV(VZ),HB(HZ
)                                           EJ
•901 REM ** ASSIGN  HORIZONTAL VALUES       IA
•902 X=24:B=0:FOR I=0 TO HZ:HH(I)=X:HB(I)
=B                                          ML
•903 X=X+2:IF X>255 THEN X=X-256:B=1        DP
•904 NEXT                                   IA
•905 REM ** ASSIGN VERTICAL VALUES          LF
•906 X=53:FOR I=0 TO VZ:VV(I)=X:X=X+8:NEX
T                                           JN
•919 REM ** KEYPRESS MATRIX                 EO
•920 DIM KP(7),K(2)                         JA
•921 FOR I=0 TO 7:KP(I)=2:NEXT:KP(1)=1:KP
(2)=0:KP(7)=3                               ML
•922 K(0)=-1:K(1)=1:K(2)=0                  BK
•929 REM ** INITIAL SPRITE POSITIONING      IO
•930 HP=79:VP=VZ:XK=1                       MH
•931 POKE VT(0),VV(VP):POKE HT(0),HH(HP):
POKE HR,HB(HP)                              AC
•939 REM ** TELL VIC-2 WHERE TO FIND SPRI
TE SHAPE                                    HK
•940 POKE LT(0),ST(XK)                      HF
•949 REM ** CREATE SCREEN DISPLAY           ED
•950 PRINT "[CLEAR]";:FOR I=0 TO 5:FOR J=
0 TO DS:PRINT "[DOWN]";:NEXT                IF
•955 PRINT MID$(WS$,1+INT(19*RND(9)),40);
:NEXT:PRINT "[HOME]";                       OK
•959 REM ** CLEAR SCREEN LINE LINK TABLE    KP
•960 L1=PEEK(217)OR 128:L2=PEEK(218)OR 12
8:POKE 217,L1:POKE 218,L2                   DN
•969 REM ** CLEAR COLLISION REGISTERS       GC
•970 Z=PEEK(CF):Z=PEEK(CS)                  IA
•989 REM ** SET CLOCK                       EI
•990 TI$="[6"0"]"                           NI
•996 RETURN                                 IM
```

BASIC SPRITES

```
•1 REM    *** BASIC SPRITES ***             GC
•2 REM A SHIP AND A ROCKET DEMONSTRATE SP
RITE CREATION AND MOVEMENT                  MN
•9 REM       *** SET-UP SUBROUTINES ***     OF
•10 GOSUB 600:REM (VIDEO MEMORY)            II
•11 GOSUB 700:REM (BACKGROUND)              AA
•12 GOSUB 800:REM (SPRITE SHAPES)           PB
•13 GOSUB 900:REM (SPRITE POSITIONS)        IE
•15 GOTO 100                                CF
•98 REM       *** ACTION LOOP ***           KJ
•100 VP=VP+1:IF VP>VZ THEN VP=0             BH
•110 HP=HP-1:IF HP<0 THEN HP=HZ             GG
•120 POKE VT(1),VV(VP)                      LO
•130 POKE HT(0),HH(HP):POKE HR,HB(HP):POK
E HT(1),HH(HP)                              LP
•140 WV=WV-1:IF WV<1 THEN WV=8              EG
•150 PRINT DD$MID$(ST$,WV,40)"[HOME]";      CI
•196 GOTO 100                               CF
•198 REM      *** WAIT FOR RESPONSE ***     HO
•296 RETURN                                 IM
•298 REM      *** ENDING ROUTINES ***       NL
•300 PRINT "[CLEAR]FINAL SCORE:  "RT*100+
VP*10:PRINT:PRINT:VP=15:POKE VT(0),VV(VP
)                                           NM
•310 PRINT "TO PLAY AGAIN, PRESS P":PRINT
 "TO QUIT, PRESS Q":HP=79                   JE
•315 HP=79:POKE HR,HB(HP):POKE HT(0),HH(H
P):XP=1:POKE LT(0),ST(XP)                   JM
•320 Z=PEEK(203):IF Z=62 THEN 370           EA
•325 POKE HT(0),HH(HP):POKE HR,HB(HP)       AO
•330 HP=HP+1:IF HP>HZ THEN HP=0             FJ
•345 IF Z<>41 THEN 320                      PE
•350 EG=0:DL=0:DT=1:DS=5:RT=0:RC=0:GOSUB
930:GOTO 100                                AK
•369 REM ** PUT VIDEO MEMORY BACK TO FIRS
T BLOCK, AND SCREEN MEMORY TO 1024          EG
•370 POKE 56578,PEEK(56578)OR3:POKE 56576
,(PEEK(56576)AND 252)OR 3                   MB
```

```
·371 POKE 53272,20:POKE 648,4 OM
·379 REM ** REENABLE SHIFT/COMMODORE AND
 RUN-STOP/RESTORE FB
·380 POKE 657,0:POKE 792,71:POKE 808,237 PK
·390 END IC
·598 REM    *** ARRANGE MEMORY *** KB
·599 REM USE FIRST VIDEO BLOCK (0 TO 1638
 3) SO NOTHING FANCY HAS TO BE DONE. ME
·600 VB=0 HP
·601 REM ** SCREEN MEMORY STAYS WHERE IT
 IS LK
·608 REM    *** REGISTER ADDRESSES *** EJ
·609 REM ** SPRITE SHAPE LOCATION TABLE IF
·610 LT(0)=2040:FOR I=1 TO 7:LT(I)=LT(I-1
 )+1:NEXT HP
·611 REM ** SPRITE COLOR TABLE LA
·612 CT(0)=53287:FOR I=1 TO 7:CT(I)=CT(I-
 1)+1:NEXT AN
·613 REM ** SPRITE HORIZONTAL POSITION TA
 BLE (LOW BYTES) CH
·614 HT(0)=53248:FOR I=1 TO 7:HT(I)=HT(I-
 1)+2:NEXT KC
·615 REM ** SPRITE VERTICAL POSITION TABL
 E IJ
·616 VT(0)=53249:FOR I=1 TO 7:VT(I)=VT(I-
 1)+2:NEXT AJ
·617 REM ** SPRITE HORIZONTAL HIGH-BIT RE
 GISTER IG
·618 HR=53264 FB
·619 REM ** SPRITE ENABLE REGISTER AF
·620 ES=53269 FE
·621 REM ** VERTICAL EXPANSION REGISTER (
 1=DOUBLE HEIGHT) IB
·622 VE=53271 GC
·623 REM ** HORIZONTAL EXPANSION REGISTER
  (1=DOUBLE WIDTH) PL
·624 HE=53277 FC
·625 REM ** SPRITE PRIORITY REGISTER (1=S
 PRITE IS IN FRONT OF FOREGROUND) GM
·626 PR=53275 FJ
·627 REM ** MULTICOLOR ENABLE REGISTER (1
 =MULTI-COLOR ENABLED) IM
·628 EM=53276 EK
·629 REM ** SPRITE MULTICOLOR COLOR REGIS
 TERS OH
·630 MR=53285:REM ('01' REGISTER: ADD 1 T
 O MR FOR '11' REGISTER) LJ
·631 REM ** SPRITE-SPRITE COLLISION REGIS
 TER OF
·632 CS=53278 GA
·633 REM ** SPRITE-FOREGROUND COLLISION R
 EGISTER MD
·634 CF=53279 FK
·635 REM ** SET-BIT AND CLEAR-BIT VALUES BI
·636 BS(0)=1:FOR I=1 TO 7:BS(I)=2*BS(I-1)
 :NEXT NE
·637 FOR I=0 TO 7:BC(I)=255-BS(I):NEXT LB
·638 REM    *** INITIALIZE VALUES *** DK
·639 REM ** FOREGROUND COLOR FL
·640 POKE 53281,6:PRINT "[CLEAR][BLUE]";:
 REM (BLUE) CC
·641 REM ** BACKGROUND COLOR DK
·642 POKE 53281,14:REM (LIGHT BLUE) AH
·643 REM ** BORDER COLOR BE
·644 POKE 53280,0:REM (BLACK) II
·645 REM ** SPRITE COLORS (DEFAULTS: WHI,
 RED,L-GRN,PUR,GRN,BLU,YEL,M-GRAY) IE
·646 POKE CT(0),0:POKE CT(1),2:REM (THE S
 HIP IS BLACK, THE ROCKET RED) IG
·647 REM ** SET PRIORITY EF
·648 POKE PR,255:REM (ALL BEHIND) JD
·649 REM ** SET HORIZONTAL SIZES BJ
·650 POKE HE,1:REM (SHIP IS DOUBLE WIDTH,
  ROCKET IS NORMAL WIDTH) PP
·651 REM ** SET VERTICAL SIZES CC
·652 POKE VE,2:REM (ROCKET IS DOUBLE HEIG
 HT) NG
·653 REM ** ENABLE SPRITES OD
·654 POKE ES,BS(0) OR BS(1):REM (SPRITES
 0 AND 1) LP
·696 RETURN IM
·698 REM    *** SET UP OCEAN *** JG
·700 R$=CHR$(18):RX$=CHR$(146):B$=" " BJ
·701 FD$=CHR$(162)+CHR$(185)+CHR$(175)+CH
 R$(228) HJ
·703 FU$=B$+CHR$(228)+CHR$(175)+CHR$(185) BF
·705 S$=R$:FOR I=1 TO 40:S$=S$+B$:NEXT:S$
 =S$+RX$ BI
·706 ST$=FU$+FD$+FU$+FD$+FU$+FD$:ST$=ST$+
 ST$ LC
·707 DD$="[HOME][20"[DOWN]"]" EL
·708 WV=8 JO
·710 PRINT "[CLEAR]"DD$MID$(ST$,WV,40)S$S
 $S$LEFT$(S$,40)"[HOME]"RX$; EM
·711 POKE 2023,160 BH
·796 RETURN IM
·798 REM    *** SPRITE SHAPES *** CD
·799 REM ** NUMBER OF SPRITE SHAPES (-1);
 NUMBER OF LINES PER SPRITE SHAPE (-1) CE
·800 NS=1 ID
·801 REM ** LOCATE SPRITE SHAPE MEMORY OM
·802 ST(0)=13:ST(1)=14 OI
·803 REM ** LOOPS HL
·804 FOR I=0 TO NS:FOR J=0 TO 20:READ SS$
 :FOR K=0 TO 2:X=0:Y=VB+ST(I)*64+J*3+K IG
·805 REM ** CONVERT STRINGS TO SHAPE BYTE
 S (USE BS(0-7) TO SET BITS) LA
·806 FOR L=1 TO 8:M=L+8*K:M=ASC(MID$(SS$,
 M,1)):IF M=42 THEN X=X OR BS(8-L) KC
·807 NEXT:POKE Y,X:NEXT:NEXT:NEXT HP
·809 REM SPRITE SHAPE DATA, 21 LINES  PER
  SHAPE: *='ON'  .='OFF' MJ
·810 DATA "[9"."]*[14"."]" AG
·811 DATA "[9"."]*[14"."]" AG
·812 DATA "[9"."]*..**[10"."]" AG
·813 DATA "[9"."]*..**[10"."]" AG
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
·814 DATA "[8"."]**..**[10"."]"              IK
·815 DATA "[8"."]**.[3"*"][10"."]"           LO
·816 DATA "[9"."][5"*"][10"."]"              AG
·817 DATA "[7"."][9"*"][8"."]"               FG
·818 DATA "[24"*"]"                          LC
·819 DATA "[24"*"]"                          LC
·820 DATA "..[20"*"].."                      HC
·821 DATA "[3"."][18"*"][3"."]"              MC
·822 DATA "[3"."][18"*"][3"."]"              MC
·823 DATA "[3"."][19"*"].."                  NG
·824 DATA "[4"."][18"*"].."                  HC
·825 DATA "[5"."][17"*"].."                  EG
·826 DATA "[24"."]"                          PC
·827 DATA "[24"."]"                          PC
·828 DATA "[24"."]"                          PC
·829 DATA "[24"."]"                          PC
·830 DATA "[24"."]"                          PC
·831 REM ** 2ND SPRITE SHAPE                 AB
·832 DATA "[21"."]*.."                       AG
·833 DATA "[21"."]*.."                       AG
·834 DATA "[20"."][3"*"]."                   LG
·835 DATA "[20"."][3"*"]."                   LG
·836 DATA "[20"."][3"*"]."                   LG
·837 DATA "[20"."][3"*"]."                   LG
·838 DATA "[20"."][3"*"]."                   LG
·839 DATA "[20"."][3"*"]."                   LG
·840 DATA "[20"."]*.*."                      PC
·841 DATA "[20"."]*.*."                      PC
·842 DATA "[24"."]"                          PC
·843 DATA "[24"."]"                          PC
·844 DATA "[24"."]"                          PC
·845 DATA "[24"."]"                          PC
·846 DATA "[24"."]"                          PC
·847 DATA "[24"."]"                          PC
·848 DATA "[24"."]"                          PC
·849 DATA "[24"."]"                          PC
·850 DATA "[24"."]"                          PC
·851 DATA "[24"."]"                          PC
·852 DATA "[24"."]"                          PC
·853 DATA "[24"."]"                          PC
·896 RETURN                                  IM
·898 REM       *** SPRITE POSITIONS ***      KA
·899 REM ** POSSIBLE POSITIONS DIM'ED        MK
·900 VZ=32:HZ=362:DIM HH(HZ),VV(VZ),HB(HZ
)                                            OD
·901 REM ** ASSIGN  HORIZONTAL VALUES        IA
·902 X=0:B=0:FOR I=0 TO HZ:HH(I)=X:HB(I)=
B                                            IJ
·903 X=X+1:IF X>255 THEN X=X-256:B=3         HC
·904 NEXT                                    IA
·905 REM ** ASSIGN VERTICAL VALUES           LF
·906 X=202:FOR I=0 TO 9:VV(I)=X:X=X-1:NEX
T:FOR I=10 TO 14:VV(I)=X:X=X-2:NEXT          LA
·907 :FOR I=15 TO 18:VV(I)=X:X=X-3:NEXT:F
OR I=19 TO 21:VV(I)=X:X=X-6:NEXT             CN
·908 FOR I=22 TO VZ:VV(I)=X:X=X-12:NEXT      FA
·929 REM ** INITIAL SPRITE POSITIONING       IO
·930 HP=160:VP=0                             LG
·931 POKE VT(0),VV(VP):POKE HT(0),HH(HP):
POKE HR,HB(HP)                               AC
·932 POKE VT(1),VV(VP):POKE HT(1),HH(HP)     CB
·939 REM ** TELL VIC-2 WHERE TO FIND SPRI
TE SHAPES                                    KK
·940 POKE LT(0),ST(0):POKE LT(1),ST(1)       FK
·996 RETURN                                  IM
```

TUMBLING DICE & DATA STRUCTURES*

FROM PAGE 37

DICE SIMULATOR

```
·1 REM                                       JD
·2 REM -- DICE SIMULATOR --                  BA
·3 REM  RUPERT REPORT #21                    OI
·4 REM                                       JD
·5 REM RANDOMLY SIMULATE THE                 EL
·6 REM OUTCOMES FOR ROLLING THREE DICE       JE
·7 REM                                       JD
·8 REM >>DETERMINE THE PROBABILITY THAT      CD
·9 REM   THE SUM IS ELEVEN                   MD
·10 N=1 : R$=CHR$(18) :REM RVS ON            ME
·20 PRINT CHR$(147);                         JL
·30 PRINT TAB(15)R$;"<< # ROLLS "            DO
·40 PRINT TAB(15)R$;"<< # SUCCESS "          AG
·50 PRINT TAB(15)R$;"<< % PROBABILITY "      BO
·60 A=INT(RND(0)*6)+1                        PI
·70 B=INT(RND(0)*6)+1                        AL
·80 C=INT(RND(0)*6)+1                        AO
·90 TTL=A+B+C                                LK
·100 IF TTL=11 THEN SXCS=SXCS+1              KO
·110 PRINT CHR$(19);N : PRINT SXCS           JM
·120 PROB=100*SXCS/N : PRINT PROB            JG
·130 N=N+1                                   CJ
·140 GOTO 60                                 PG
```

DICE ANALYZER

```
·1 REM                                       JD
·2 REM -- DICE ANALYZER --                   OE
·3 REM  RUPERT REPORT #21                    OI
·4 REM                                       JD
·5 REM ENUMERATE ALL POSSIBLE                FL
·6 REM OUTCOMES FOR ROLLING THREE DICE       JE
·7 REM                                       JD
·8 REM >>CALCULATE THE PROBABILITY THAT      NA
·10 REM    1) 3 ONES ARE ROLLED              NP
·11 REM    2) AT LEAST 1 THREE AND 1 SIX
              ARE ROLLED                     HM
·12 REM    3) NO TWOS ARE ROLLED             HF
·13 REM    4) THE SUM IS ELEVEN              DN
·14 REM                                      JD
```

* *Bug Repellent* line codes at right of program lines are for C-64 only! See VIC codes on next page.

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
•15 REM >ON THE NTH ROLL (N = 1 TO 216):  DF
•16 REM ROLL(N,Y) = DIE Y'S VALUE          DF
•17 REM CT(N,Z) = # DICE WITH VALUE Z      KK
•18 REM TTL(N) = SUM OF DICE VALUES        AH
•19 REM                                    JD
•20 DIM ROLL(216,3),CT(216,6),TTL(216),RN
 UM(216)                                   JN
•30 N=1 : M=1 : PRINT CHR$(147)            HH
•40 FOR A=1 TO 6 : FOR B=1 TO 6            FJ
•50 FOR C=1 TO 6                           ID
•60 PRINT A;B;C                            OD
•70 ROLL(N,1)=A:ROLL(N,2)=B:ROLL(N,3)=C    OL
•80 CT(N,A)=CT(N,A)+1 :CT(N,B)=CT(N,B)+1   HH
•90 CT(N,C)=CT(N,C)+1                      NP
•100 TTL(N)=A+B+C                          IA
•110 N=N+1 : NEXT C : NEXT B : NEXT A      BN
•120 PRINT"CALCULATING [3"."] "            JP
•130 FOR N=1 TO 216                        AE
•140 :IF CT(N,1)=3 THEN T1=T1+1            LI
•150 :IF CT(N,3)>=1 AND CT(N,6)>=1 THEN T
 2=T2+1                                    EA
•160 :IF CT(N,2)=0 THEN T3=T3+1            FE
•170 :IF TTL(N)=11 THEN T4=T4+1:RNUM(M)=N
 :M=M+1                                    LE
•180 NEXT N                                NA
•190 PRINT                                 JJ
•200 PRINT,"# SUCCESS"," % PROBABILITY"    JO
•210 PRINT "TEST 1",T1,T1*100/216          OL
•220 PRINT "TEST 2",T2,T2*100/216          EM
•230 PRINT "TEST 3",T3,T3*100/216          OF
•240 PRINT "TEST 4",T4,T4*100/216          NO
•250 PRINT                                 JJ
•260 PRINT"SUCCESSFUL ROLLS FOR TEST 4:"   HD
•270 FOR M=1 TO T4                         OG
•280 RM=RNUM(M)                            CB
•290 PRINT ROLL(RM,1);ROLL(RM,2);ROLL(RM,
 3),                                       LA
•300 NEXT M                                NB
```

VIC 20 BUG REPELLENT LINE CODES FOR TUMBLING DICE & DATA STRUCTURES

DICE SIMULATOR

# 1:	JD	# 7:	JD	# 40:	LK	# 100:	GC
# 2:	PB	# 8:	KJ	# 50:	ME	# 110:	NO
# 3:	OE	# 9:	LG	# 60:	PI	# 120:	CM
# 4:	JD	# 10:	MI	# 70:	AL	# 130:	CJ
# 5:	KL	# 20:	JH	# 80:	AO	# 140:	PD
# 6:	MK	# 30:	FO	# 90:	LK		

DICE ANALYZER (16K EXPANSION REQUIRED!)

# 1:	JD	# 7:	JD	# 14:	JD	# 20:	EG
# 2:	IK	# 8:	JI	# 15:	IC	# 30:	HA
# 3:	OE	# 10:	OI	# 16:	IH	# 40:	EP
# 4:	JD	# 11:	MM	# 17:	IE	# 50:	HK
# 5:	EF	# 12:	FC	# 18:	PO	# 60:	OF
# 6:	MK	# 13:	GD	# 19:	JD	# 70:	OL

# 80:	EA	# 140:	IL	# 200:	GN	# 260:	MG
# 90:	NP	# 150:	JO	# 210:	NK	# 270:	OM
# 100:	IA	# 160:	IB	# 220:	KH	# 280:	CB
# 110:	AJ	# 170:	FF	# 230:	BA	# 290:	NG
# 120:	EL	# 180:	NG	# 240:	KN	# 300:	NH
# 130:	ON	# 190:	JJ	# 250:	JJ		

ADDRESSING THE COMMODORE, PART II

FROM PAGE 77

(See article for instructions on entering!)

THE QUEST

```
10 ;
20 ;THE QUEST
30 ;
40  *=8000 ;OR 'ORG EQU $8000' (OR $2000
)
50 ;
60 BUFLEN=23
70 CHROUT=$FFD2
80 ;
90  JMP BEGIN
100 ;
110 TEXT .BYTE 87,72,69,82,69,32,73,83
120  .BYTE 32,84,72,69,32,67,79,77
130  .BYTE 77,79,68,79,82,69,63
140 ;
150 BEGIN LDX #0
160 ;
170 LOOP LDA TEXT,X
180  JSR CHROUT
190  INX
200  CPX #BUFLEN
210  BNE LOOP
220  RTS
```

RESPONSE

```
10 ;
20 ; RESPONSE
30 ;
40  *=$8000 ;OR 'ORG EQU $8000' (OR $200
0)
50 ;
60 EOL=13 ;END-OF-LINE CHARACTER
70 BUFLEN=40 ;LENGTH OF TEXT BUFFER
80 FILLCH=$20 ;ASCII CODE FOR A SPACE
90 CHROUT=$FFD2
100 ;
110  JMP START
120 ;
130 TEXT .BYTE 'YOU CAN FIND HIM IN 64K'
,13
140 ;
150 ;CLEAR TEXT BUFFER
160 ;
170 START LDA #FILLCH
```

```
180  LDX #BUFLEN
190 STUFF DEX
200  STA TXTBUF,X
210  BNE STUFF
220 ;
230 ;STORE MESSAGE IN BUFFER
240 ;
250  LDX #0
260 LOOP1 LDA TEXT,X
270  STA TXTBUF,X
280  CMP #EOL
290  BEQ PRINT
300  INX
310  CPX #BUFLEN
320  BCC LOOP1
330 ;
340 ;PRINT MESSAGE
350 ;
360 PRINT LDX #0
370 LOOP2 LDA TXTBUF,X
380  PHA
390  JSR CHROUT
400  PLA
410  CMP #EOL
420  BNE NEXT
430  JMP FINI
440 NEXT INX
450  CPX #BUFLEN
460  BCC LOOP2
470 ;
480 FINI
490  RTS
500 ;
510 TXTBUF=*
520  *=*+BUFLEN
530 ;
540  .END
```

SUPERHERO

FROM PAGE 17

```
·10 REM     SUPERHERO                                  FG
·20 REM     BY J.C.HILTY                               LE
·30 REM     1036 BARNSLEY DR.                          JK
·40 REM     LIBRARY,PA.,15129                          KF
·100 PRINT"[CLEAR]"                                    HH
·102 SR=0:ALT=000                                      OP
·103 Q=192:L=1141:C=28:CM=54272:M=0                    FO
·105 V=53248:O=54272                                   HC
·108 POKE 52,48:POKE 56,48                             IC
·110 POKE 53281,6:POKE 53280,2                         DD
·120 PRINT"[DOWN][DOWN]"TAB(5)"[RVSON][RE
D][7" "][c *]"                                         CH
·130 PRINT TAB(5)"[RVSON][8" "][c *]"                  NL
·140 PRINT TAB(5)"[RVSON][BLACK] [RED] [Y
ELLOW][7" "][RED] [c *][RVSOFF] [RVSON][
BLACK] [RED] [c *][BLACK] [RED]  [c *][B
LACK] [RED]  [c *][BLACK] [RED]  [c *]"                CE
·150 PRINT TAB(5)"[RVSON][BLACK]  [YELLOW
][7" "][BLACK] [RED] [c *][BLACK]  [RED]
 [BLACK]  [RED][3" "][BLACK] [RED][3" "]
[BLACK] [RED]  [c *]"                                  DP
·160 PRINT TAB(5)"[RVSON][BLACK]  [YELLOW
][5" "][BLACK][3" "][YELLOW]  [BLACK]  [
YELLOW]  [BLACK] [YELLOW][3" "][BLACK] [
YELLOW][3" "][BLACK] [YELLOW][3" "]"                   JG
·170 PRINT TAB(5)"[BLACK][c *][RVSON] [YE
LLOW][7" "][BLACK] [YELLOW]  [BLACK]  [Y
ELLOW]  [BLACK] [YELLOW] [BLACK] [YELLOW
] [BLACK] [YELLOW]  [BLACK]  [YELLOW] [B
LACK] [YELLOW] "                                       IA
·180 PRINT TAB(6)"[BLACK][c *][RVSON][YEL
LOW][7" "][BLACK] [YELLOW]  [BLACK]  [YE
LLOW]  [BLACK] [YELLOW][3" "][BLACK] [YE
LLOW] [RVSOFF] [RVSON][BLACK]  [YELLOW][
3" "][BLACK][c *]"                                     IB
·190 PRINT TAB(10)"[RVSON][BLACK]  [YELLO
W]  [BLACK] [YELLOW]  [BLACK]  [YELLOW]
 [BLACK] [YELLOW] [RVSOFF] [BLACK][c *][
RVSON] [YELLOW][3" "][RVSOFF][BLACK][c *
][RVSON][YELLOW] [RVSOFF] [BLACK][c *][R
VSON][YELLOW] "                                        GP
·200 PRINT TAB(5)"[RVSON][BLACK] [RED][4"
 "][BLACK]  [YELLOW]  [BLACK] [YELLOW][6
" "][RVSOFF][BLACK][c *][RVSON][YELLOW]
"                                                      FG
·210 PRINT TAB(5)"[RVSON][BLACK]  [RED][4
" "][BLACK] [YELLOW]  [RVSOFF][BLACK][c
*][RVSON][YELLOW][6" "]"                               EG
·220 PRINT TAB(5)"[BLACK][c *][RVSON] [YE
LLOW][7" "]"                                           AI
·230 PRINT TAB(6)"[BLACK][c *][RVSON][YEL
LOW][7" "][RVSOFF] [RVSON][BLACK] [RED][
c *][BLACK] [RED][c *][RVSOFF] [RVSON][B
LACK] [RED][3" "][c *][RVSOFF] [RVSON][B
LACK] [RED]  [c *][RVSOFF]  [RVSON][BLAC
K] [RED]  [c *]"                                       CL
·240 PRINT TAB(15)"[RVSON][BLACK] [YELLOW
] [BLACK] [YELLOW] [RVSOFF] [RVSON][BLAC
K] [YELLOW][4" "][RVSOFF] [RVSON][BLACK]
 [YELLOW][3" "][RVSOFF]  [RVSON][BLACK]
[YELLOW][3" "]"                                        FA
·250 PRINT TAB(15)"[RVSON][BLACK] [YELLOW
] [BLACK] [YELLOW] [RVSOFF] [RVSON][BLAC
K] [YELLOW] [BLACK] [c *][RVSOFF]  [RVSO
N] [YELLOW] [BLACK] [YELLOW] [RVSOFF]  [
RVSON][BLACK] [YELLOW] [BLACK] [YELLOW]
"                                                      LE
·260 PRINT TAB(15)"[RVSON][BLACK] [YELLOW
][3" "][RVSOFF] [RVSON][BLACK] [YELLOW][
3" "][RVSOFF]  [RVSON][BLACK] [YELLOW][3
" "][RVSOFF]  [RVSON][BLACK] [YELLOW] [B
LACK] [YELLOW] "                                       CI
·270 PRINT TAB(15)"[RVSON][BLACK] [YELLOW
] [BLACK] [YELLOW] [RVSOFF] [RVSON][BLAC
K] [YELLOW] [BLACK]  [c *][RVSOFF] [RVSO
```

```
N][BLACK] [YELLOW] [BLACK][RVSOFF][c *][
RVSON][YELLOW]  [RVSOFF] [RVSON][BLACK]
[YELLOW] [BLACK] [YELLOW] "                HA
•280 PRINT TAB(15)"[BLACK][c *][RVSON][YE
LLOW] [RVSOFF][BLACK][c *][RVSON][YELLOW
] [RVSOFF] [BLACK][c *][RVSON][YELLOW][4
" "][RVSOFF] [BLACK][c *][RVSON][YELLOW]
 [RVSOFF] [BLACK][c *][RVSON][YELLOW] [R
VSOFF] [BLACK][c *][RVSON][YELLOW][3" "]
"                                          NN
•290 PRINT                                 JJ
•300 PRINT TAB(9)"SAVE THE SPACE SHUTTLE"  LL
•310 PRINT TAB(16)"PRESS ANY KEY"          KL
•320 GET A$:IF A$="" THEN 320              HI
•330 PRINT"[HOME]":PRINT"[21"[DOWN]"]"  T
AB(15)"PLEASE WAIT[3"."]"                  JI
•340 PRINT TAB(15)"READING DATA"           GH
•370 FOR X=12288 TO 12350                  GB
•380 READ A:POKE X,A:NEXT                  BA
•390 FOR X=12352 TO 12414                  DG
•400 READ A:POKE X,A:NEXT                  BA
•410 FOR X= 12416 TO 12478                 FD
•420 READ A:POKE X,A:NEXT                  BA
•430 FOR X=12480 TO 12542                  DC
•440 READ A:POKEX,A:NEXT                   BA
•464 FOR X=828 TO 973                      KC
•466 READ A:POKEX,A:NEXT                   BA
•470 PRINT"[CLEAR]"                        HH
•480 POKE 53280,0:POKE 53281,0             BH
•490 POKE 2040,194:POKE 2041,195           IP
•500 POKE V+39,6:POKE V+40,1               DA
•510 POKE V+0,70:POKE V+1,80:POKE V+2,70:
POKE V+3,140                               FI
•530 POKE V+29,3:POKE V+23,3               MB
•540 POKE V+28,1:POKE V+37,10:POKE V+38,2  OD
•550 PRINT"[HOME]":PRINT"[DOWN][CYAN]"TAB
(12)"GRAPHICS[3" "]CHART"                  AF
•560 POKE V+21,3                           FC
•570 PRINT"[5"[DOWN]"]"TAB(12)"[EP][6" "]
SUPERHERO"                                 JH
•580 PRINT"[6"[DOWN]"]"TAB(18)"SPACESHUTT
LE"                                        CA
•590 PRINT"[4"[DOWN]"]"TAB(12)"[GREEN]*[C
YAN][5" "]ZYPTONITE  METEOR"               CH
•600 PRINT"[DOWN]"TAB(14)"PRESS ANY KEY"   FF
•610 GET A$:IF A$="" THEN 610              IC
•620 PRINT"[CLEAR]":POKE V+21,0            AH
•630 POKE 53280,2:POKE 53281,0             EF
•700 DIMA(3)                               EJ
•710 A(0)=2:A(1)=-80:A(2)=-2:A(3)=80       LN
•720 WL=160:HL=32:SC=1024:A=SC+81          AJ
•740 FOR I=1 TO 6                          JB
•741 PRINT        "[RVSON][GREEN][15" "][RV
SOFF][7" "][RVSON][17" "]"                 FF
•742 NEXT I                                MN
•744 FOR I=1 TO 9                          JC
•745 PRINT         "[RVSON][39" "]"        GO
•746 NEXT I                                MN
•748 FOR I=1 TO 8                          JD
•750 PRINT"[RVSON][15" "][RVSOFF][7" "][R
VSON][17" "]"                              PM
•760 NEXT I                                MN
•770 POKEA,4                               DL
•780 J=INT(RND(1)*4):X=J                   MO
•790 B=A+A(J):IF PEEK(B)=WLTHENPOKEB,J:PO
KEA+A(J)/2,HL:A=B:GOTO 780                 BP
•800 J=(J+1)*-(J<3):IF J<>XTHEN790         FD
•810 J=PEEK(A):POKE A,HL:IFJ<4THENA=A-A(J
):GOTO 780                                 HB
•820 PRINT"[HOME]":PRINT"[15"[DOWN]"]"TAB
(16)"[CYAN]SCORE"                          DN
•830 PRINT TAB(17);SR                      LI
•831 PRINT"[4"[DOWN]"]"TAB(16)"ALT":PRINT
 TAB(16);ALT                               MF
•832 POKE 2040,192                         DH
•833 POKE V+0,150:POKE V+1,60:POKE V+2,15
0:POKE V+3,190                             OK
•834 POKE V+21,3                           FC
•840 POKE 1113,81 :POKE 1113+CM,7          PJ
•842 POKE 1835,81:POKE 1835+CM,7           JF
•844 POKE 1825,81:POKE 1825+CM,7           AH
•846 POKE 1509,81:POKE 1509+CM,7           EP
•848 POKE 1599,81:POKE 1599+CM,7           EF
•850 POKE 1605,81:POKE 1605+CM,7           PL
•852 POKE 1137,81:POKE 1137+CM,7           AF
•854 POKE 1131,81:POKE 1131+CM,7           GJ
•856 POKE 1857,81:POKE 1857+CM,7           JF
•858 POKE 1851,81:POKE 1851+CM,7           GJ
•860 POKE 1141,28:POKE 1141+CM,3           FE
•865 POKE A,42:J=2                         OC
•870 Q=Q+1:IF Q=195 THEN Q=192             AK
•871 POKE 2040,Q                           OH
•872 ALT=ALT+1                             FP
•873 IF ALT=800THEN3500                    DP
•875 PRINT"[HOME]":PRINT"[22"[DOWN]"][CYA
N]" TAB(16);ALT                            AJ
•876 B=A+A(J)/2                            JM
•877 IF PEEK(B)=32THENPOKEB,42:POKEA,32:A
=B:J=(J+2)+4*(J>1)                         LG
•878 IF PEEK(B)=81 THEN POKE B,42:POKE A,
81:POKE A+CM,7:A=B:J=(J+2)+4*(J>1)         KJ
•879 IF PEEK(B)=28 THEN 3000               IG
•880 J=(J-1)-4*(J=0)                       HI
•881 SYS(828):R=PEEK(254)-PEEK(253)        ML
•890 IF R=0 THEN 870                       AO
•900 W=PEEK(L+R)                           GN
•910 IF W=81 THEN 1500                     HI
•920 IF W<>32 THEN 870                     PH
•930 L=L+R:POKE L-R,32:POKE(L-R)+CM,5      BN
•940 POKE L,C:POKE L+CM,3                  BB
•950 GOTO 870                              CP
•1500 M=M+1:IF M=10 THEN 2500              LH
•1505 L=L+R:POKE L-R,32                    BD
•1510 POKEL,C:POKEL+CM,3                   BB
•1520 GOSUB 2000                           FL
•1530 PRINT"[HOME]":SR=SR+10:PRINT"[16"[D
OWN]"][CYAN]"TAB(17);SR                    HG
•1540 GOTO 870                             CP
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
2000 FOR X=O TO O+24:POKE X,0:NEXT                CE
2010 POKE O,150:POKEO+1,200:POKEO+5,8:PO
KEO+6,248                                         NL
2020 POKEO+24,15:POKEO+4,17                       GB
2030 FOR T=0TO50:NEXT                             IM
2040 POKEO+4,16                                   GC
2050 FOR T=0TO50:NEXT                             IM
2060 POKEO+24,0                                   GH
2070 RETURN                                       IM
2100 FOR X=O TO O+24:POKEX,0:NEXT                 CH
2110 POKEO+24,15:POKEO+12,160:POKEO+13,2
52                                                JO
2120 POKEO+8,80:POKEO+7,40:POKE O+11,129          FB
2130 FOR T=0TO100:NEXT                            LC
2140 POKEO+11,128:RETURN                          HA
2500 PRINT"[CLEAR]"                               HH
2502 PRINT"[CYAN][15"[DOWN]"]"TAB(6)"CON
GRATULATIONS"                                     GK
2503 PRINT"[DOWN][6" "]PLAY ON!"                  KF
2504 FOR T=0 TO 3000:NEXT                         PL
2510 M=0                                          DG
2520 SR=SR+100:ALT=ALT -200                       NI
2525 L=1141                                       LN
2528 PRINT"[CLEAR]"                               HH
2530 GOTO 710                                     CA
3000 PRINT"[CLEAR]"                               HH
3005 FOR T=1 TO 30                                NI
3010 POKE V+39,2:POKE V+39,1:POKE V+39,6          LL
3020 NEXT                                         IA
3030 GOSUB 2100                                   FE
3040 PRINT"[14"[DOWN]"]SORRY,A ZYPTONITE
 METEOR CRUSHED YOU."                             NE
3050 FOR T=0 TO 3000:NEXT                         PL
3060 GOTO 4000                                    FF
3500 PRINT "[CLEAR]"                              HH
3510 FOR T=1 TO 20                                MF
3520 POKE V+40,2:POKE V+40,6:POKE V+40,1          FD
3530 NEXT                                         IA
3540 GOSUB 2100                                   FE
3550 PRINT"[12"[DOWN]"][CYAN]THE SHUTTLE
 IS 800 MILES UP--TOO LATE."                      KF
3560 FOR T=0 TO 3000:NEXT                         PL
3570 GOTO 4000                                    FF
4000 PRINT"[CLEAR]"                               HH
4005 POKE V+21,0                                  FF
4010 PRINT"[12"[DOWN]"][RED]S C O R E  "
;SR                                               OM
4020 PRINT"[DOWN][DOWN]PLAY AGAIN[3"-"]Y
 OR N "                                           DK
4030 GET JUNK$:IF JUNK$<>""THEN 4030              FG
4040 GET A$:IF A$=""THEN 4040                     LO
4050 IF A$="Y" THEN 4060                          MF
4055 END                                          IC
4060 PRINT"[CLEAR]"                               HH
4070 M=0                                          DG
4080 L=1141                                       LN
4090 ALT=000                                      CK
4100 SR=0                                         IO
4110 GOTO 710                                     CA
6240 REM MULTICOLOR SUPERHERO X3                  OF
6250 DATA 0,160,0,0,144,0,0,144,0,0,192,
0,3,160,0,12,160,0                                IG
6260 DATA 48,160,0,48,160,0,48,160,0,48,
128,0,192,128,0,192,192,0                         MJ
6270 DATA 192,192,0,192,128,0,192,128,0,
0,128,0,0,128,0,0,192,0                           AN
6280 DATA 0,192,0,0,192,0,0,240,0                 FI
6290 DATA 0,160,0,0,144,0,0,144,0,0,192,
0,3,160,0,14,160,0,50,170,64,50,170,64            EI
6300 DATA 50,160,0,56,128,0,204,128,0,19
6,192,0,192,192,0,194,32,0                        KF
6310 DATA 194,32,0,8,8,0,8,8,0,48,12,0,4
8,12,0,12,12,0,0,15,0                             IN
6320 DATA 0,160,0,0,144,0,0,144,0,0,192,
0,3,160,0,14,160,64,48,171,0,48,162,0             HN
6330 DATA 48,164,0,48,136,0,192,128,0,19
2,192,0,192,192,0,194,32,0                        MO
6340 DATA 194,32,0,2,32,0,2,32,0,2,32,0,
3,48,0,3,48,0,3,252,0                             GA
6350 REM SPACESHUTTLE                             EA
6360 DATA 0,0,0,0,0,0,32,0,0,32,0,0,48,0
,0,56,0,0                                         OJ
6370 DATA 63,255,192,63,255,224,61,85,48
,63,255,56                                        OJ
6380 DATA 63,255,252,12,62,0,8,60,0,8,56
,0,0,48,0                                         GG
6390 DATA 0,32,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0                                            CG
6450 REM ML JOYSTICK ROUTINE                      PM
6460 DATA 173,1,220,74,176,40,74,74,176,
11,169,0                                          FB
6470 DATA 133,254,169,41,133,253,76,182,
3,74,176,11,169,0,133,254,169,39                  NM
6480 DATA 133,253,76,182,3,169,0,133,254
,169,40,133,253,76,182,3,74,176                   CA
6490 DATA 39,74,176,11,169,39,133,254,16
9,0,133,253,76,182,3,74,176,11                    KH
6500 DATA 169,41,133,254,169,0,133,253,7
6,182,3,169,40,133,254,169,0                      NF
6510 DATA 133,253,76,182,3,74,176,11,169
,0,133,254,169,1,133,253,76,182,3                 LG
6520 DATA 74,176,11,169,1,133,254,169,0,
133,253,76,182,3,169,0,133,254                    CH
6530 DATA 133,253,173,1,220,74,74,74,74,
74,176,7,169,1,133,80,76,203,3                    LJ
6540 DATA 169,0,133,80,96,234,234                 EM
```

AHOY! DOCK

FROM PAGE 30

```
0 PRINT"[CLEAR]"                                  HH
1 DIMSU$(500),TA$(500),AN$(500),MN$(500)
,ID$(500),CM$(500):P$=",":OPEN15,8,15             IB
2 GOTO100                                         CF
```

```
·10 PRINT"[CLEAR]":PRINTTAB(12);"ONE MOME
NT READING":PRINT:PRINTTAB(20-LEN(Z$)/2)
;Z$                                          FI
·15 OPEN5,8,5,"0:"+Z$+",S,R":INPUT#15,AX,
BX$,CX,DX$:IFAX<>0THENGOSUB9000:RETURN       KB
·18 FORI=1TO500:C=I                          FN
·20 INPUT#5,SU$(I),TA$(I),AN$(I),MN$(I),I
D$(I),CM$(I)                                 DM
·30 IFSU$(I)="EOF"THENCLOSE5:C=C-1:RETURN EM
·40 NEXT                                     IA
·50 PRINT:PRINT"40 CHARACTERS MAXIMUM":FO
RI=1TO4000:NEXT:PRINT"[CLEAR]":RETURN        OJ
·60 PRINT:PRINT"80 CHARACTER MAXIMUM":FOR
I=1TO4000:NEXT:PRINT"[CLEAR]":RETURN         NB
·80 OPEN5,8,5,"@0:"+Z$+",S,W":FORI=1TOC      HN
·82 PRINT#5,SU$(I):PRINT#5,TA$(I):PRINT#5
,AN$(I):PRINT#5,MN$(I):PRINT#5,ID$(I)        NF
·83 PRINT#5,CM$(I):NEXT:I=C+1                FM
·84 SU$(I)="EOF":TA$(I)="TITLE":AN$(I)="A
UTHOR":MN$(I)="MAG":ID$(I)="ISSUE"           JO
·86 CM$(I)="COMMENT":PRINT#5,SU$(I):PRINT
#5,TA$(I):PRINT#5,AN$(I):PRINT#5,MN$(I)      JP
·87 PRINT#5,ID$(I):PRINT#5,CM$(I)            IK
·88 CLOSE5:RETURN                            JK
·100 PRINT"[CLEAR]":POKE53280,6:POKE53281
,0:FORI=1TO40:PRINT"[RED][s S][WHITE][s
S]";:NEXT:PRINT"[RED][s S][WHITE][s S]"; KI
·110 PRINT" [RVSON][c A][32"[s C]"][c S][
RVSOFF] ";                                   FP
·120 PRINT "[RED][s S][WHITE][s S]";:PRIN
T"[RED][s S][WHITE][s S] [RVSON][s B][RV
SOFF][32"="][RVSON][s B][RVSOFF] [RED][s
 S][WHITE][s S]";                            HJ
·130 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON] [3"*"]AHOY DO
CK FILE[3"*"] [RVSOFF][5"="][RVSON][s B]
[RVSOFF] [RED][s S][WHITE][s S]";            HD
·140 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][22" "][RVSOFF
][5"="][RVSON][s B][RVSOFF] [RED][s S][W
HITE][s S]";                                 ND
·150 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][3" "]ADD[4" "
]-/- RECORD  [RVSOFF][5"="][RVSON][s B][
RVSOFF] [RED][s S][WHITE][s S]";             AC
·160 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][3" "]SEARCH -
/- RECORD  [RVSOFF][5"="][RVSON][s B][RV
SOFF] [RED][s S][WHITE][s S]";               HG
·170 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][3" "]DELETE -
/- RECORD  [RVSOFF][5"="][RVSON][s B][RV
SOFF] [RED][s S][WHITE][s S]";               IP
·180 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][3" "]MODIFY -
/- RECORD  [RVSOFF][5"="][RVSON][s B][RV
SOFF] [RED][s S][WHITE][s S]";               HM
·190 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][3" "]SORT[3"
"]-/- RECORD  [RVSOFF][5"="][RVSON][s B]
[RVSOFF] [RED][s S][WHITE][s S]";            OE
·200 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][3" "]LIST[3"
"]-/- RECORD  [RVSOFF][5"="][RVSON][s B]
[RVSOFF] [RED][s S][WHITE][s S]";            MO
·210 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][3" "]PRINT  -
/- RECORD  [RVSOFF][5"="][RVSON][s B][RV
SOFF] [RED][s S][WHITE][s S]";               MM
·215 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][5"="][RVSON][22" "][RVSOFF
][5"="][RVSON][s B][RVSOFF] [RED][s S][W
HITE][s S]";                                 ND
·220 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF][32"="][RVSON][s B][RVSOFF]
 [RED][s S][WHITE][s S]";                    DF
·230 PRINT"[RED][s S][WHITE][s S] [RVSON]
[c Q][32"[s C]"][c W][RVSOFF] [RED][s S]
[WHITE][s S]";                               IF
·240 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF]";:PRINTCHR$(159);:PRINT"[R
VSON] F1[3"."]ADD[4" "]";:PRINTCHR$(5);  PG
·250 PRINT"[6"="]";:PRINTCHR$(159);:PRINT
" [RVSON]F2[3"."]SEARCH ";:PRINTCHR$(31)
;:                                           MP
·260 PRINT"[WHITE][RVSON][s B][RVSOFF] [R
ED][s S][WHITE][s S]";:PRINT"[RED][s S][
WHITE][s S] [RVSON][s B][RVSOFF]";:PRINT
CHR$(159);:PRINT"[RVSON] F3[3"."]DELETE
";                                           BM
·270 PRINTCHR$(5);:PRINT"[6"="]";:PRINTCH
R$(159);:PRINT"[RVSON] F4[3"."]MODIFY "; BJ
·280 PRINTCHR$(31);:PRINT"[WHITE][RVSON][
s B][RVSOFF] [RED][s S][WHITE][s S]";:PR
INT"[RED][s S][WHITE][s S] [RVSON][s B][
RVSOFF]";:PRINTCHR$(159);                    EO
·290 PRINT"[RVSON] F5[3"."]SORT[3" "]";:P
RINTCHR$(5);:PRINT"[6"="]";:                 IL
·300 PRINTCHR$(159);:PRINT" [RVSON]F6[3".
"]LIST[3" "]";:PRINTCHR$(31);:PRINT"[WHI
TE][RVSON][s B][RVSOFF] [RED][s S][WHITE
][s S][WHITE]";                              LJ
·310 PRINT"[RED][s S][WHITE][s S] [RVSON]
[s B][RVSOFF]";:PRINTCHR$(159);:PRINT"[R
VSON] F7[3"."]PRINT  ";:PRINTCHR$(5);:   CE
·320 PRINT"[6"="]";:PRINTCHR$(159);:PRINT
" [RVSON]F8[3"."]END[4" "]";:PRINTCHR$(3
1);:                                         CL
·330 PRINT"[WHITE][RVSON][s B][RVSOFF] [R
ED][s S][WHITE][s S]";:PRINT"[RED][s S][
WHITE][s S] [RVSON][c Z][32"[s C]"][c X]
[RVSOFF]";                                   GJ
·340 PRINT" [RED][s S][WHITE][s S]";:FORI
=1TO40:PRINT"[RED][s S]";:PRINT"[WHITE][
s S]";:NEXT                                  EG
·350 GETRE$:IFRE$=""THEN350                  AE
```

```
·360 IFRE$=CHR$(140)THEN680                         PL
·400 FORI=133TO140:IFRE$=CHR$(I)THEN520             GK
·410 NEXT                                           IA
·420 PRINT"[CLEAR]":POKE53281,0:POKE53280
 ,0:PRINT"[WHITE]":PRINTTAB(12);"INVALID
 RESPONSE"                                          OL
·430 PRINT:PRINTTAB(7);"PLEASE PRESS FUNC
 TION KEY":RE$="":FORI=1TO4000:NEXT                 FD
·440 GOTO100                                        CF
·450 DATA++++++++++++++++++++++,+...SELE
 CTION MENU....+,+                        +         BN
·460 DATA + 1--BUSINESS          +,+ 2--DI
 SK OPERATION   +,+ 3--EDUCATIONAL      +           AM
·470 DATA + 4--FUN AND GAMES     +,+ 5--GR
 APHICS         +,+ 6--HARDWARE         +           BM
·480 DATA + 7--HOME USE          +,+ 8--LA
 NGUAGES-MLX    +,+ 9--MISCELLANEOUS    +           IL
·490 DATA + 10-MUSICAL           +,+ 11-SI
 MULATIONS      +,+ 12-SOFTWARE         +           EI
·500 DATA + 13-TUTORIAL          +,+ 14-UT
 ILITIES        +,+                     +           BJ
·510 DATA+....PLEASE SELECT....+,++++++++
 +++++++++++++++                                    PN
·520 PRINT"[CLEAR]":PRINT:POKE53280,7:POK
 E53281,1:PRINT"[BLACK]":FORI=1TO20:READM
 $                                                  DE
·530 PRINTTAB(9);M$:NEXT:RESTORE:INPUT"[9
 "[RIGHT]"]";SE$:SE=VAL(SE$)                        KG
·532 IFSE$="/"THEN440                               ME
·535 IFSE<1ORSE>14THEN520                           LO
·540 IFSE$="1"THENZ$="BUSINESS"                     AP
·550 IFSE$="2"THENZ$="DISK OPERATION"               EC
·560 IFSE$="3"THENZ$="EDUCATIONAL"                  IG
·570 IFSE$="4"THENZ$="FUN & GAMES"                  FA
·580 IFSE$="5"THENZ$="GRAPHICS"                     OE
·590 IFSE$="6"THENZ$="HARDWARE"                     LC
·600 IFSE$="7"THENZ$="HOME USE"                     KE
·610 IFSE$="8"THENZ$="LANGUAGES-MLX"                GC
·620 IFSE$="9"THENZ$="MISCELLANEOUS"                AI
·630 IFSE$="10"THENZ$="MUSICAL"                     PJ
·640 IFSE$="11"THENZ$="SIMULATIONS"                 CK
·650 IFSE$="12"THENZ$="SOFTWARE"                    BP
·660 IFSE$="13"THENZ$="TUTORIAL"                    FJ
·670 IFSE$="14"THENZ$="UTILITIES"                   EI
·680 RE=ASC(RE$):ON(RE-132)GOSUB1000,2000
 ,3000,4000,5000,6000,7000,8000                     DH
·690 Z$="XX":DE$="":DE=.:K1=.:GOTO100               EM
·700 REM*********************************
 *                                                  EO
·710 REM* WRITTEN BY GLENN LUMPKINS                 LF
·720 REM* FEB. 1985                                 KD
·730 REM*********************************
 *                                                  EO
·999 REM=====ADD ROUTINE=====                       OC
·1000 PRINT"[CLEAR]":GOSUB10:PRINT"[CLEAR
 ]":C=C+1                                           HO
·1010 IFK=1THENK=0:RETURN                           GN
·1020 PRINTTAB(8);"[RVSON]: SUBJECT  OF
 ARTICLE :[RVSOFF]":PRINT"[BLUE]";:INPUTS
 U$(C)                                              AC
·1025 IFSU$(C)="/"THENRETURN                        DD
·1030 IFLEN(SU$(C))>79THENGOSUB60:PRINT:
 GOTO1020                                           PF
·1040 PRINT"[BLACK]":PRINTTAB(8);"[RVSON]
 : TITLE[4" "]OF  ARTICLE :[RVSOFF]":PRIN
 T"[BLUE]";:INPUTTA$(C)                             FA
·1045 IFTA$(C)="/"THEN1020                          GG
·1050 IFLEN(TA$(C))>39THENGOSUB50:PRINT:G
 OTO1040                                            LH
·1060 PRINT"[BLACK]":PRINTTAB(8);"[RVSON]
 : AUTHOR'S[8" "]NAME :[RVSOFF]":PRINT"[B
 LUE]";:INPUTAN$(C)                                 IJ
·1065 IFAN$(C)="/"THEN1040                          GC
·1070 IFLEN(AN$(C))>39THENGOSUSB50:PRINT:
 GOTO1060                                           MO
·1080 PRINT"[BLACK]":PRINTTAB(8);"[RVSON]
 :[SS]NAME[5" "]OF MAGAZINE :[RVSOFF]":PR
 INT"[BLUE]";:INPUTMN$(C)                           MM
·1085 IFMN$(C)="/"THEN1060                          JM
·1090 IFLEN(MN$(C))>39THENGOSUB50:PRINT:G
 OTO1080                                            AN
·1100 PRINT"[BLACK]":PRINTTAB(8);"[RVSON]
 :ISSUE # / PAGE / DATE :[RVSOFF]":PRINT"
 [BLUE]";:INPUTID$(C)                               AO
·1105 IFID$(C)="/"THEN1080                          HI
·1110 IFLEN(ID$(C))>39THENGOSUSB50:PRINT:
 GOTO1100                                           GH
·1120 PRINT"[BLACK]":PRINTTAB(8);"[RVSON]
 : COMMENTS  OR[3" "]NOTES :[RVSOFF]":PRI
 NT"[BLUE]";:INPUTCM$(C)                            IP
·1130 IFLEN(CM$(C))>79 THENGOSUB60:PRINT:
 GOTO1120                                           CO
·1135 IFCM$(C)="/"THEN1100                          HI
·1140 PRINT"[BLACK]":PRINTTAB(5);"->PRESS
  ANY KEY TO CONTINUE<-"                            EG
·1150 GETDE$:IFDE$=""THEN1150                       BO
·1160 PRINT"[CLEAR]":PRINTTAB(20-LEN(Z$)/
 2);Z$:PRINT                                        BG
·1170 PRINTTAB(5);"[RVSON][C][RVSOFF]ONTI
 NUE OR [RVSON][W][RVSOFF]RITE TO DISK"             OF
·1180 GETDE$:IFDE$=""THEN1180                       CD
·1185 IFDE$="/"THENRETURN                           IO
·1190 IFDE$<>"C"ANDDE$<>"W"THEN1180                 DO
·1200 IFDE$="C"THENPRINT"[CLEAR]":C=C+1:G
 OTO1020                                            BL
·1205 PRINT:PRINTTAB(15);"WRITING TO":PRI
 NT:PRINTTAB(20-LEN(Z$)/2);Z$                       ED
·1210 GOSUB80:RETURN                                MF
·1999 REM====DELETE ROUTINE=======                  DN
·2000 PRINT"[CLEAR]":PRINT:PRINT:PRINT              PH
·2005 PRINT"  DELETE [RVSON][E][RVSOFF]NT
 IRE FILE OR [RVSON][O][RVSOFF]NE RECORD"           GE
·2010 GETDE$:IFDE$=""THEN2010                       BO
·2015 IFDE$="/"THENRETURN                           IO
·2020 IFDE$<>"E"ANDDE$<>"O"THEN2000                 FI
·2030 IFDE$="E"THENGOSUB9200:RETURN                 EP
```

```
·2040 FL=1:GOSUB7000:FL=.:RETURN           JD
·2999 REM=====SORT PART1======             PK
·3000 GOSUB10:PRINT"[CLEAR]":FORI=1TO5:PR
 INT:NEXT:PRINTTAB(15);"SORT BY":PRINT     JE
·3010 F1=1:GOSUB5010:IFDE$="/"THENRETURN   OF
·3020 IFDE$="S"THENK1=1:GOSUB9300:GOTO307
 0                                         ED
·3030 IFDE$="T"THENK1=2:GOSUB9300:GOTO307
 0                                         ED
·3040 IFDE$="A"THENK1=3:GOSUB9300:GOTO307
 0                                         GP
·3050 IFDE$="M"THENK1=4:GOSUB9300:GOTO307
 0                                         LK
·3060 IFDE$="I"THENK1=5:GOSUB9300:GOTO307
 0                                         IF
·3070 K1=.:PRINT"[CLEAR]":PRINTTAB(8);"ON
 E MOMENT WRITING SORTED":PRINT            OO
·3080 PRINTTAB(20-LEN(Z$)/2);Z$            KL
·3090 GOSUB80:RETURN                       MF
·3999 REM=====PRINT RECORD/FILE=====       LP
·4000 PRINT"[CLEAR]":PRINTTAB(8);"[RVSON]
 [E][RVSOFF]NTIRE OR [RVSON][O][RVSOFF]NE
  RECORD"                                  LD
·4010 GETDE$:IFDE$=""THEN4010              BM
·4020 IFDE$="/"THENRETURN                  IO
·4030 IFDE$<>"E"ANDDE$<>"O"THEN4010        GB
·4040 IFDE$="O"THENK2=1:GOSUB7000:IFK2=.T
 HENRETURN                                 NI
·4050 IFDE$="E"THENGOSUB10:GOTO4200        JH
·4060 OPEN3,4:XX$="[s T][17"="][s Y]":PRI
 NT#3,Z$;" RECORD NUMBER";I:PRINT#3        ML
·4070 PRINT#3:PRINT#3,XX$:PRINT#3,TAB(3);
 "S U B J E C T":PRINT#3                   FH
·4080 PRINT#3,SU$(I):PRINT#3:PRINT#3,XX$:
 PRINT#3,TAB(5);"T I T L E":PRINT#3        HD
·4090 PRINT#3,TA$(I):PRINT#3:PRINT#3,XX$:
 PRINT#3,TAB(4);"A U T H O R":PRINT#3      JC
·4100 PRINT#3,AN$(I):PRINT#3:PRINT#3,XX$:
 PRINT#3,TAB(2);"M A G A Z I N E"          PG
·4110 PRINT#3:PRINT#3,MN$(I):PRINT#3:PRIN
 T#3,XX$:PRINT#3,TAB(5);"I S S U E"        NH
·4120 PRINT#3:PRINT#3,ID$(I):PRINT#3:PRIN
 T#3,XX$:PRINT#3,TAB(3);"C O M M E N T"    BJ
·4130 PRINT#3:PRINT#3,CM$(I):CLOSE3:RETUR
 N                                         FL
·4199 REM=====PRINT ENTIRE RECORD=====     GL
·4200 PRINT"[CLEAR]":PRINTTAB(14);"NOW PR
 INTING":PRINT:PRINTTAB(20-LEN(Z$)/2);Z$   OJ
·4210 XX$="[s T][17"="][s Y]":OPEN3,4:PRI
 NT:PRINT"RECORD NUMBER"                   AH
·4230 PRINT#3,Z$;" FILE":PRINT#3           GH
·4240 FORI=1TOC:PRINTI:PRINT#3:PRINT#3,"=
 =>RECORD NUMBER";I;" <==":PRINT#3         AA
·4250 PRINT#3,XX$:PRINT#3,TAB(3);"S U B J
  E C T":PRINT#3:PRINT#3,SU$(I):PRINT#3    PD
·4260 PRINT#3,XX$:PRINT#3,TAB(5);"T I T L
 E":PRINT#3:PRINT#3,TA$(I):PRINT#3         PE
·4270 PRINT#3,XX$:PRINT#3,TAB(4);"A U T H
```

```
 O R":PRINT#3:PRINT#3,AN$(I):PRINT#3       ME
·4280 PRINT#3,XX$:PRINT#3,TAB(2);"M A G A
  Z I N E":PRINT#3:PRINT#3,MN$(I)          EO
·4290 PRINT#3:PRINT#3,XX$:PRINT#3,TAB(5);
 "I S S U E":PRINT#3:PRINT#3,ID$(I)        BP
·4300 PRINT#3:PRINT#3,XX$:PRINT#3,TAB(3);
 "C O M M E N T":PRINT#3:PRINT#3,CM$(I)    MI
·4310 PRINT#3:NEXT:CLOSE3:RETURN           GC
·4999 REM=====SEARCH ROUTINE=====          AA
·5000 PRINT"[CLEAR]":FORI=1TO5:PRINT:NEXT
 :PRINTTAB(15);"SEARCH BY":PRINT           HN
·5010 PRINT:PRINTTAB(15);"[RVSON][S][RVSO
 FF]UBJECT":PRINT:PRINTTAB(15);"[RVSON][T
 ][RVSOFF]ITLE":PRINT                      CG
·5020 PRINTTAB(15);"[RVSON][A][RVSOFF]UTH
 OR":PRINT:PRINTTAB(15);"[RVSON][M][RVSOF
 F]AGAZINE":PRINT                          EK
·5030 PRINTTAB(15);"[RVSON][I][RVSOFF]SSU
 E                                         BL
·5040 GETDE$:IFDE$=""THEN5040              EC
·5045 IFDE$="/"THENRETURN                  IO
·5050 IFDE$<>"S"ANDDE$<>"T"ANDDE$<>"A"AND
 DE$<>"M"ANDDE$<>"I"THEN5000               IC
·5055 IFF1=1THENF1=.:RETURN                KC
·5060 IFDE$="S"THENPRINT"[CLEAR]":FORI=1T
 O4:PRINT:NEXT:GOTO5090                    PL
·5070 IFDE$="T"THENPRINT"[CLEAR]":FORI=1T
 O4:PRINT:NEXT:GOTO5110                    AH
·5075 IFDE$="A"THENPRINT"[CLEAR]":FORI=1T
 O4:PRINT:NEXT:GOTO5130                    OM
·5080 IFDE$="M"THENPRINT"[CLEAR]":FORI=1T
 O4:PRINT:NEXT:GOTO5150                    GC
·5085 IFDE$="I"THENPRINT"[CLEAR]":FORI=1T
 O4:PRINT:NEXT:GOTO5170                    FA
·5090 PRINT"[CLEAR]":PRINTTAB(13);"ENTER
 ";:PRINT"[BLUE]";:PRINT"[SUBJECT]":PRINT
 "[BLACK]"                                 GG
·5100 INPUTS$:GOSUB10:GOTO5300             LH
·5110 PRINT"[CLEAR]":PRINTTAB(13);"ENTER
 ";:PRINT"[BLUE]";:PRINT"[TITLE]":PRINT"[
 BLACK]"                                   FN
·5120 INPUTT$:GOSUB10:GOTO5300             NE
·5130 PRINT"[CLEAR]":PRINTTAB(13);"ENTER
 ";:PRINT"[BLUE]";:PRINT"[AUTHOR]":PRINT"
 [BLACK]"                                  LO
·5140 INPUTA$:GOSUB10:GOTO5300             JB
·5150 PRINT"[CLEAR]":PRINTTAB(12);"ENTER
 ";:PRINT"[BLUE]";:PRINT"[MAGAZINE]":PRIN
 T"[BLACK]"                                AL
·5160 INPUTM$:GOSUB10:GOTO5300             NF
·5170 PRINT"[CLEAR]":PRINTTAB(13);"ENTER
 ";:PRINT"[BLUE]";:PRINT"[ISSUE]":PRINT"[
 BLACK]"                                   JM
·5180 INPUTI$:GOSUB10:GOTO5300             MJ
·5300 PRINT"[CLEAR]":PRINT" #";TAB(10);"O
 NE MOMENT SEARCHING":PRINTTAB(20-LEN(Z$)
 /2);Z$                                    FE
·5310 FORI=1TOC                            JM
```

```
•5320 IFDE$="S"ANDLEFT$(S$,LEN(S$))=LEFT$
 (SU$(I),LEN(S$))THENPRINTI;SU$(I):K=K+1  JB
•5330 IFDE$="T"ANDLEFT$(T$,LEN(T$))=LEFT$
 (TA$(I),LEN(T$))THENPRINTI;TA$(I):K=K+1  LF
•5340 IFDE$="I"ANDLEFT$(I$,LEN(I$))=LEFT$
 (ID$(I),LEN(I$))THENPRINTI;ID$(I):K=K+1  FF
•5350 IFDE$="M"ANDLEFT$(M$,LEN(M$))=LEFT$
 (MN$(I),LEN(M$))THENPRINTI;MN$(I):K=K+1  BB
•5360 IFDE$="A"ANDLEFT$(A$,LEN(A$))=LEFT$
 (AN$(I),LEN(A$))THENPRINTI;AN$(I):K=K+1  KB
•5370 IFK=15THENK=0:PRINT:PRINTTAB(4);"CO
 RRECT RECORD FOUND [RVSON][Y][RVSOFF] OR
 [RVSON][N][RVSOFF]":Y=1                  GG
•5380 IFY=1THENGETRE$:IFRE$=""THEN5380    NN
•5385 IFRE$="/"THENRETURN                  LM
•5390 IFY=1ANDRE$<>"N"ANDRE$<>"Y"THEN5380 DK
•5400 IFRE$="N"THENPRINT"[CLEAR]":Y=.:RE$
 ="":NEXT:GOTO5440                        KI
•5410 IFRE$="Y"THENY=.:GOTO5500           NI
•5415 IFC=IANDK=>1THENK=15:GOTO5370       AE
•5420 NEXT:PRINT                           FO
•5430 PRINTTAB(11);"RECORD NOT FOUND":PRI
 NT                                        PA
•5440 PRINTTAB(10)"[RVSON][A][RVSOFF]BORT
 OR [RVSON][R][RVSOFF]ETRY                GN
•5450 GETDE$:IFDE$=""THEN5450             BN
•5460 IFDE$="/"ORDE$="A"THENRETURN        EK
•5470 IFDE$="R"THENDE$="":GOTO5000        FO
•5480 IFDE$<>"A"ANDDE$<>"R"THEN5450       PF
•5500 PRINT:PRINTTAB(13);"SELECT RECORD":
 PRINT"[BLUE]";:INPUTRN:IFRN>CTHEN5500    KB
•5510 PRINT"[BLACK]":PRINT"[CLEAR]":I=RN:
 FA=1:GOTO7010                            DD
•5999 REM====MODIFY RECORD======          LN
•6000 F4=1:GOSUB7000                       PP
•6010 PRINTTAB(7);"[BLACK]";"[RVSON][1. S
 UBJECT OF ARTICLE ][RVSOFF]":PRINT"[BLUE
 ]";SU$(I):PRINT                          CN
•6020 PRINTTAB(7);"[BLACK]";"[RVSON][2. T
 ITLE OF ARTICLE[3" "]][RVSOFF]":PRINT"[B
 LUE]";TA$(I):PRINT                       AB
•6030 PRINTTAB(7);"[BLACK]";"[RVSON][3. A
 UTHOR'S NAME[6" "]][RVSOFF]":PRINT"[BLUE
 ]";AN$(I):PRINT                          AN
•6040 PRINTTAB(7);"[BLACK]";"[RVSON][4. N
 AME OF MAGAZINE[3" "]][RVSOFF]":PRINT"[B
 LUE]";MN$(I):PRINT                       PM
•6050 PRINTTAB(7);"[BLACK]";"[RVSON][5. I
 SSUE #/PAGE/DATE  ][RVSOFF]":PRINT"[BLUE
 ]";ID$(I):PRINT                          DO
•6060 PRINTTAB(7);"[BLACK]";"[RVSON][6. C
 OMMENT OR NOTE[4" "]][RVSOFF]":PRINT"[BL
 UE]";CM$(I):PRINT                        BG
•6070 PRINTTAB(7);"[BLACK]";"[RVSON][7. W
 RITE TO DISK[6" "]][RVSOFF]":PRINT       OC
•6080 PRINTTAB(7);"[3"="]>CHANGE WHICH LI
 NE<=="                                   BO
•6090 GETDE$:IFDE$=""THEN6090             EK
•6100 DE=VAL(DE$):IFDE$="/"THENRETURN     NL
•6110 IFDE<1ORDE>7THEN6090                LO
•6120 IFDE=7THENGOSUB80:RETURN            HO
•6130 IFDE=1THENPRINT"[CLEAR]":PRINTTAB(5
 );"[RVSON]: SUBJECT OF ARTICLE :[RVSOFF]
 ":PRINT:PRINT"[BLUE]";                   II
•6140 IFDE=1THENPRINTSU$(I):PRINT:PRINT"E
 NTER NEW VALUE":PRINT:INPUTSU$(I)        NH
•6150 IFLEN(SU$(I))>79THENGOSUB60:PRINT:G
 OTO6130                                  GM
•6155 IFSU$(I)="/"THENRETURN              GN
•6160 IFDE=2THENPRINT"[CLEAR]":PRINTTAB(5
 );"[RVSON]: TITLE OF ARTICLE :[RVSOFF]":
 PRINT:PRINT"[BLUE]";                     JG
•6170 IFDE=2THENPRINTTA$(I):PRINT:PRINT"E
 NTER NEW VALUE":PRINT:INPUTTA$(I)        MC
•6180 IFLEN(TA$(I))>39THENGOSUB50:PRINT:G
 OTO6160                                  DN
•6185 IFTA$(I)="/"THENRETURN              GO
•6190 IFDE=3THENPRINT"[CLEAR]":PRINTTAB(5
 );"[RVSON]: AUTHOR'S NAME :[RVSOFF]":PRI
 NT:PRINT"[BLUE]";                        HE
•6200 IFDE=3THENPRINTAN$(I):PRINT:PRINT"E
 NTER NEW VALUE":PRINT:INPUTAN$(I)        FL
•6210 IFLEN(AN$(I))>39THENGOSUB50:PRINT:G
 OTO6190                                  OI
•6215 IFAN$(I)="/"THENRETURN              DA
•6220 IFDE=4THENPRINT"[CLEAR]":PRINTTAB(5
 );"[RVSON]: NAME OF MAGAZIZNE :[RVSOFF]"
 :PRINT:PRINT"[BLUE]";                    EE
•6230 IFDE=4THENPRINTMN$(I):PRINT:PRINT"E
 NTER NEW VALUE":PRINT:INPUTMN$(I)        KG
•6240 IFLEN(MN$(I))>39THENGOSUB50:PRINT:G
 OTO6220                                  GM
•6245 IFMN$(I)="/"THENRETURN              EE
•6250 IFDE=5THENPRINT"[CLEAR]":PRINTTAB(5
 );"[RVSON]: ISSUE / PAGE / DATE :[RVSOFF
 ]":PRINT:PRINT"[BLUE]";                  IA
•6260 IFDE=5THENPRINTID$(I):PRINT:PRINT"E
 NTER NEW VALUE":PRINT:INPUTID$(I)        JB
•6270 IFLEN(ID$(I))>39THENGOSUB50:PRINT:G
 OTO6250                                  DP
•6275 IFID$(I)="/"THENRETURN              DG
•6280 IFDE=6THENPRINT"[CLEAR]":PRINTTAB(5
 );"[RVSON]: COMMENTS OR NOTES :":PRINT:P
 RINT"[BLUE]";                            BL
•6290 IFDE=6THENPRINTCM$(I):PRINT:PRINT"E
 NTER NEW VALUE":PRINT:INPUTCM$(I)        AO
•6300 IFLEN(CM$(I))>70THENGOSUB60:PRINT,G
 OTO6280                                  DP
•6305 IFCM$(I)="/"THENRETURN              CF
•6310 PRINT"[CLEAR]":GOTO6010             PO
•6999 REM====LIST + DELETE ROUTINE ======
 =                                        CE
•7000 PRINT"[CLEAR]":GOSUB10:PRINT"[CLEAR
 ]":FORI=1TOC:PRINT:PRINT                 OB
•7010 PRINTTAB(9);"[RVSON][:SUBJECT OF AR
 TICLE: ][RVSOFF]":PRINT"[BLUE]";SU$(I):PR
```

```
INT                                            BM
·7020 PRINTTAB(9);"[BLACK][RVSON][:TITLE
 OF ARTICLE  :][RVSOFF]":PRINT"[BLUE]";TA
 $(I):PRINT                                    MJ
·7030 PRINTTAB(9);"[BLACK][RVSON][:AUTHOR
 'S NAME[5" "]:][RVSOFF]":PRINT"[BLUE]";A
 N$(I):PRINT                                   MI
·7040 PRINTTAB(9);"[BLACK][RVSON][:NAME O
 F MAGAZINE  :][RVSOFF]":PRINT"[BLUE]";MN
 $(I):PRINT                                    PI
·7050 PRINTTAB(9);"[BLACK][RVSON][:ISSUE
 #/PAGE/DATE :][RVSOFF]":PRINT"[BLUE]";ID
 $(I):PRINT                                    BD
·7060 PRINTTAB(9);"[BLACK][RVSON][:COMMEN
 T OR NOTES  :][RVSOFF]":PRINT"[BLUE]";CM
 $(I):PRINT                                    OC
·7065 IFFA=1THENPRINT:PRINT"[BLACK]";TAB(
 8);"PRESS ANY KEY TO CONTINUE"                AH
·7070 IFFA=1THENGETDE$:IFDE$=""THEN7070        BK
·7075 IFDE$="/"THENRETURN                      IO
·7080 IFFA=1THENFA=.:RETURN                    MC
·7090 IFFL=1THENPRINT"[BLACK]";TAB(5);"DE
 LETE THIS RECORD [RVSON][Y][RVSOFF] OR [
 RVSON][N][RVSOFF]"                            HM
·7100 IFFL=1THENGETDE$:IFDE$=""THEN7100        AH
·7110 IFFL=1ANDDE$="Y"THEN7180                 DI
·7120 IFFL=1THENPRINT"[CLEAR]":NEXTI:RETU
 RN                                            ML
·7125 IFF4=1THENPRINT"[BLACK]";TAB(5);"MO
 DIFY THIS RECORD [RVSON][Y][RVSOFF] OR [
 RVSON][N][RVSOFF]":GOTO7140                   AH
·7130 PRINT"[BLACK]";TAB(7);"CONTINUE PAG
 ING [RVSON][Y][RVSOFF] OR [RVSON][N][RVS
 OFF]"                                         AF
·7140 GETDE$:IFDE$=""THEN7140                  EB
·7145 IFDE$="/"THENRETURN                      IO
·7150 IFDE$<>"Y"ANDDE$<>"N"THEN7140            CF
·7155 IFF4=1ANDDE$="N"THENPRINT"[CLEAR]":
 NEXT:RETURN                                   EB
·7160 IFDE$="N"THENRETURN                      LD
·7165 IFF4=1ANDDE$="Y"THENF4=.:PRINT"[CLE
 AR]":RETURN                                   EC
·7170 PRINT"[CLEAR]":NEXT:K2=.:RETURN          KJ
·7180 SU$(I)="DELETE":OPEN5,8,5,"@0:"+Z$+
 ",S,W":PRINT"[CLEAR]"                         OF
·7190 PRINTTAB(4);"ONE MOMENT DELETING RE
 CORD #";I:FORI=1TOC                           CP
·7200 IFSU$(I)="DELETE"THENC=C-1:NEXT          BI
·7210 PRINT#5,SU$(I)P$TA$(I)P$AN$(I)P$MN$
 (I)P$ID$(I)P$CM$(I)                           KH
·7220 NEXT:I=C+1:SU$(I)="EOF":TA$(I)="TIT
 LE":AN$(I)="AUTHOR":MN$(I)="MAG"              EH
·7230 ID$(I)="ISSUE":CM$(I)="COMMENT"          OA
·7240 PRINT#5,SU$(I)P$TA$(I)P$AN$(I)P$MN$
 (I)P$ID$(I)P$CM$(I):CLOSE5:FL=.:RETURN        HB
·7999 REM====END===                            CC
·8000 PRINT"[CLEAR]":FORX=1TO12:PRINT:NEX
 T:PRINTTAB(9);"IT'S BEEN A PLEASURE"          CO
·8010 FORI=1TO15:POKE53281,I+1:FOR X=1TO2
 00:NEXTX:NEXTI:POKE53281,6                    AA
·8020 PRINT"[CLEAR]":PRINT"[HOME]":END         FO
·8999 REM====== INITIALIZE FILE======          PG
·9000 PRINT"[CLEAR]":PRINTTAB(5);"THIS FI
 LE '";Z$;"' HAS NOT BEEN INITIALIZED ":C
 LOSE5                                         PL
·9010 PRINT"OR YOU DO NOT HAVE THE CORREC
 T DISK IN  THE DRIVE !"                       PL
·9020 PRINT:PRINTTAB(8);"[RVSON][C][RVSOF
 F]ONTINUE -/- [RVSON][R][RVSOFF]ETURN"        OE
·9030 GETDE$:IFDE$=""THEN9030                  GH
·9035 IFDE$="/"THENK=1:RETURN                  PM
·9040 IFDE$="R"THENK=1:RETURN                  EF
·9050 IFDE$<>"R"ANDDE$<>"C"THEN9030            PN
·9060 OPEN5,8,5,"0:"+Z$+",S,W":PRINT"[CLE
 AR]":PRINTTAB(14);"INITIALIZING"              PE
·9070 PRINT:PRINTTAB(20-LEN(Z$)/2);Z$          GB
·9080 SU$(1)="EOF":TA$(1)="TITLE":AN$(1)=
 "AUTHOR":MN$(1)="MAG":ID$(1)="ISSUE"          BG
·9090 CM$(1)="COMMENT"                         DC
·9100 PRINT#5,SU$(1)P$TA$(1)P$AN$(1)P$MN$
 (1)P$ID$(1)P$CM$(1):CLOSE5:RETURN             PG
·9199 REM=====DELETE THE ENTIRE FILE ====
 =                                             JA
·9200 PRINT"[CLEAR]":FORI=1TO10:PRINT:NEX
 T:PRINTTAB(11);"!! ARE YOU SURE !!"           JM
·9210 PRINT:PRINTTAB(15);"[RVSON][Y][RVSO
 FF] OR [RVSON][N][RVSOFF]"                    KK
·9220 GETDE$:IFDE$=""THEN9220                  GC
·9225 IFDE$="/"THENRETURN                      IO
·9230 IFDE$<>"Y"ANDDE$<>"N"THEN9220            PC
·9240 IFDE$="N"THENRETURN                      LD
·9250 PRINT:PRINTTAB(10);"ONE MOMENT DELE
 TING":PRINT:PRINTTAB(20-LEN(Z$)/2);Z$         DO
·9260 OPEN5,8,5,"@0:"+Z$+",S,W":GOSUB9080
 :RETURN                                       BJ
·9299 REM=====SORT PART2=====                  KP
·9300 PRINT"[CLEAR]":PRINTTAB(9);"ONE MOM
 ENT NOW SORTING":PRINT                        EC
·9310 PRINTTAB(20-LEN(Z$)/2);Z$                KL
·9320 ONK1GOTO9350,9400,9450,9500,9550         EO
·9350 F=.:FORJ=1TOC-1                          MI
·9360 IFSU$(J)<=SU$(J+1)THEN9380               PO
·9370 GOSUB10000                               II
·9380 NEXT:IFF=1THEN9350                       ND
·9390 RETURN                                   IM
·9400 F=.:FORJ=1TOC-1                          MI
·9410 IFTA$(J)<=TA$(J+1)THEN9430               NA
·9420 GOSUB10000                               II
·9430 NEXT:IFF=1THEN9400                       NN
·9440 RETURN                                   IM
·9450 F=.:FORJ=1TOC-1                          MI
·9460 IFAN$(J)<=AN$(J+1)THEN9480               OJ
·9470 GOSUB10000                               II
·9480 NEXT:IFF=1THEN9450                       NE
·9490 RETURN                                   IM
·9500 F=.:FORJ=1TOC-1                          MI
```

```
•9510 IFMN$(J)<=MN$(J+1)THEN9540                NC
•9520 GOSUB10000                                PP
•9530 NEXT:IFF=1THEN9500                        NK
•9540 RETURN                                    IM
•9550 F=.:FORI=1TOC-1                           MH
•9560 IFID$(J)<=ID$(J+1)THEN9580                NK
•9570 GOSUB10000                                II
•9580 NEXT:IFF=1THEN9550                        NF
•9590 RETURN                                    IM
•10000 S1$=SU$(J):T1$=TA$(J):A1$=AN$(J):M
 1$=MN$(J):I1$=ID$(J):C1$=CM$(J)                PB
•10010 SU$(J)=SU$(J+1):SU$(J+1)=S1$:TA$(J
 )=TA$(J+1):TA$(J+1)=T1$:F=1                    OJ
•10020 AN$(J)=AN$(J+1):AN$(J+1)=A1$:MN$(J
 )=MN$(J+1):MN$(J+1)=M1$                        MO
•10030 ID$(J)=ID$(J+1):ID$(J+1)=I1$:CM$(J
 )=CM$(J+1):CM$(J+1)=C1$:RETURN                 KA
```

GO-LISTER*

FROM PAGE 73

First byte: C000   Last byte: C140   SYS to start: 49152

```
C000: A9 01 A2 04 A0 FF 20 BA CC
C008: FF A9 00 20 BD FF 20 C0 70
C010: FF A2 01 20 C9 FF A0 08 46
C018: 84 46 A0 01 84 45 4C 87 22
C020: C0 A0 00 B1 FB C9 00 F0 E9
C028: 41 C9 22 D0 06 A9 01 45 1C
C030: 43 85 43 A5 43 C9 01 F0 E0
C038: 12 B1 FB C9 89 F0 28 C9 2E
C040: 8D F0 24 C9 A7 F0 20 C9 2F
C048: CB F0 1C A0 01 20 53 C0 F6
C050: 4C 23 C0 C0 00 F0 0F A9 EA
C058: 01 18 65 FB 90 02 E6 FC 49
C060: 85 FB 88 4C 53 C0 60 4C 77
C068: CF C0 A0 02 B1 FB C9 00 13
C070: F0 14 A0 03 B1 FB 85 FD 4A
C078: C8 B1 FB 85 FE C8 20 53 AF
C080: C0 84 43 4C 23 C0 88 B1 73
C088: 45 C9 00 F0 34 C8 B1 45 7C
C090: 85 3D C8 B1 45 85 3E A0 77
C098: 00 B1 45 AA C8 B1 45 85 7F
C0A0: 46 8A 85 45 A9 00 85 FB 67
C0A8: A9 08 85 FC A9 0D 20 D2 86
C0B0: FF A5 3D AA A5 3E 20 CD 10
C0B8: BD A9 3A 20 D2 FF 4C 72 0C
C0C0: C0 A9 0D 20 D2 FF 20 CC 18
C0C8: FF A9 01 20 C3 FF 60 A0 58
C0D0: 00 84 14 84 15 C8 B1 FB 79
C0D8: C9 20 F0 F9 C9 A0 F0 F5 FE
C0E0: C9 A4 F0 F1 C9 30 30 32 8E
C0E8: C9 3A 10 2E E9 2F 85 07 D0
C0F0: A5 15 85 22 A5 14 0A 26 3D
C0F8: 22 0A 26 22 65 14 85 14 80
C100: A5 22 65 15 85 15 06 14 F6
C108: 26 15 A5 14 65 07 85 14 03
C110: 90 02 E6 15 C8 B1 FB 4C 61
C118: E4 C0 20 53 C0 A5 3D C5 9A
C120: 14 D0 13 A5 3E C5 15 D0 A7
C128: 0D A9 20 20 D2 FF A5 FD 95
C130: AA A5 FE 20 CD BD A0 00 CB
C138: B1 FB C9 2C F0 91 4C 23 CD
C140: C0 01
```

START & END FILE ADDRESS

FROM PAGE 17

```
•2 PRINT"[CLEAR][DOWN][7" "][RVSON]START
 & END FILE ADDRESS":REM BY DAVE SMART           DM
•4 FORI=828TO904:READA:POKEI,A:NEXT              KH
•6 OPEN15,8,15,"I0"                              OO
•8 INPUT"[3"[DOWN]"] FILE TO CHECK ";F$:P
RINT                                             DE
•10 "PMSD-3"              SEQ                    OG
•12 POKE251,P-S*256:POKE252,S                    CI
•14 PRINT" ";F$                                  KC
•16 OPEN1,8,2,"0:"+F$                            GO
•18 GET#1,A$,B$                                  AK
•20 D=ASC(A$+CHR$(0))+256*ASC(B$+CHR$(0)) IL
•22 INPUT#15,E,E$,E1,E2                          ED
•24 IFE>0THENPRINT:PRINT,E$:CLOSE1:CLOSE1
 5:END                                           KL
•30 PRINTTAB(15)"START ADDR.(DEC)"D              JH
•32 W=D: GOSUB54                                 HC
•34 PRINTTAB(26)"(HEX) "H$                       NJ
•36 PRINT"[DOWN]PLEASE WAIT[3"."]"               IJ
•38 SYS831                                       DN
•40 F=P:P=PEEK(251)+PEEK(252)*256:T=P            CN
•42 Z$=RIGHT$("[3" "]"+STR$(T-F),5)              DH
•44 PRINT"[UP][18" "]"Z$;" BYTES OF MEM.[
 DOWN]"                                          JP
•46 Z=VAL(Z$)                                    KH
•48 PRINTTAB(17)"END ADDR.(DEC)"Z+W:D=W+Z
 :GOSUB54                                        AN
•50 PRINTTAB(26)"(HEX) "H$                       NJ
•52 GOTO60                                       PG
•54 H$="":M=4096:N=3:IFD<256THENM=16:N=1         AL
•56 FORH=0TON:C=INT(D/M):D=D-C*M:M=M/16:C
 =C+48:IFC>57THENC=C+7                           DA
•58 H$=H$+CHR$(C):NEXT:RETURN                    BL
•60 CLOSE1:CLOSE15                               BB
•62 PRINT"[DOWN][DOWN]DO YOU WISH TO CHEC
 K ANOTHER FILE (Y/N)"                           NH
•64 GETA$:IFA$=""THEN64                          GB
•66 IFA$<>"Y"THENEND                             DI
•68 IFA$="Y"THEN6                                DG
•70 DATA 76,96,3,162,1,32,198,255,32             NH
•72 DATA 228,255,160,0,145,251,230               OB
•74 DATA 251,208,8,230,252,165,252               HA
•76 DATA 201,208,240,6,165,144,41,64             KD
```

*To enter *Go-Lister*, you must use *Flankspeed* (see page 86).

```
•78 DATA 240,231,76,204,255,162,1,32          KC
•80 DATA 201,255,162,54,134,1,160,0           DB
•82 DATA 177,251,32,210,255,230,251           MK
•84 DATA 208,2,230,252,165,251,197,253        DM
•86 DATA 208,239,165,252,197,254,208          FL
•88 DATA 233,162,55,134,1,76,204,255          EF
```

FISH MATH

FROM PAGE 29

FROM PAGE 29

```
•0 GOTO90                                      PP
•2 D=INT(RND(1)*9)+1                           OB
•3 E=INT(RND(1)*9)+1:I=D*E:PRINT"[HOME][2
 2"[DOWN]"][7"[RIGHT]"]";D;"[RVSON][s V]"
 ;E;                                           EM
•4 X=2:Q=0:FORT=1TO7:POKEU,241:FORTT=1TO3
 00:NEXT:POKEU,0:FORTT=1TO50:NEXT:NEXT:GO
 TO34                                          GC
•8 IFQ=ITHEN18                                 OG
•9 IFQ<>ITHEN70                                JO
•14 POKEA,G:POKEA+B,2:Q=Q+1:FORT=170TO135
 STEP-1:POKEU+1,T:NEXT:POKEU+1,0:POKEA,32
 :GOTO40                                       LH
•18 K=K+1:PRINT"[CLEAR][BLACK][RVSON]GREA
 T[3"!"]":PRINT"[DOWN][DOWN][RED][RVSON]C
 ORRECT ANSWER!":PRINT"[DOWN][DOWN]";D;"[
 RVSON][s V]";E;"=";I                          CF
•19 PRINT"[DOWN][PURPLE][RVSON]YOUR SCORE
  IS NOW";K:FORT=135TO241:POKEU,T:NEXT         DP
•20 POKEU,0:FORT=1TO2:POKEU,241:FORTT=1TO
 200:NEXT:POKEU,0:FORTT=1TO50:NEXT:NEXT        PF
•21 PRINT"[4"[DOWN]"][BLACK][RVSON]HIT SP
 ACE"                                          BG
•22 IFPEEK(197)<>32THEN22                      EB
•23 PP=PP+1:IFPP<10THEN29                      GO
•24 POKE36869,240:PRINT"[CLEAR][7" "]GAME
  OVER":PRINT"[3"[DOWN]"]FINAL SCORE:";K       OJ
•25 PRINT"[4"[DOWN]"]YOU HAVE COMPLETED T
 EN PROBLEMS WITHOUT GUMS GETTING YOU!"        JA
•26 PRINT"[3"[DOWN]"]HIT [RVSON]F1[RVSOFF
 ] TO PLAY[10" "]AGAIN."                       PH
•27 IFPEEK(197)<>39THEN27                      BN
•28 RUN95                                      PL
•29 PRINT"[CLEAR]":POKE36869,255               CA
•30 A=8131:B=30720:C=7680:V=36878:POKEV,1
 5:U=36876:POKEV+1,238:X=1                     NL
•31 FORT=1TO125:S=INT(RND(1)*396)+7702:PO
 KES,5:POKES+B,0:NEXT:H=7679                   ML
•33 IFX=1THEN2                                 KD
•34 POKEC+B,0:POKEC+1+B,0:POKEC,3:POKEC+1
 ,4:IFA=CORA=C+1THEN55                         HE
•35 WAIT653,1,1:IFC>8160THENPOKEC,32:POKE
 C+1,32:C=7680                                 AM
•36 POKEA,32:F=PEEK(197):IFF<>23ANDF<>31T
 HENA=A-22:G=0                                 AD
•37 IFF=31THENA=A-1:G=2                        BC
•38 IFF=23ANDA<8163THENA=A+1:G=1               FL
•39 IFPEEK(A)=5THEN14                          OI
•40 POKEA+B,2:POKEA,G:POKEC,32:POKEC+1,32
 :C=C+1:IFPEEK(C+1)=5THENC=C+22                IA
•41 IFA<HTHEN8                                 KK
•42 GOTO33                                     PG
•55 L=15                                       GD
•56 FORT=200TO135STEP-1:POKEU+1,T:NEXT:PO
 KEU+1,135:FORT=1TO50:NEXT:POKEU+1,0           CL
•57 L=L-1:POKEV,L:IFL>5THEN56                  KA
•58 FORT=1TO1500:NEXT:POKEU-2,135:FORT=1T
 O1000:NEXT:POKEU-2,0                          AL
•60 POKE36869,240:PRINT"[CLEAR]GUMS GOT Y
 A!":PRINT"[4"[DOWN]"]BUT DON'T WORRY,YOU
 [3" "]WERE DELICOUS!!"                        BI
•61 PRINT"[4"[DOWN]"]HIT[RIGHT][RVSON]F1[
 RVSOFF] TO PLAY AGAIN"                        EO
•62 IFPEEK(197)<>39THEN62                      BC
•63 RUN29                                      PO
•70 POKE36869,240:IFQ>ITHENPRINT"[CLEAR]Y
 OU ATE TOO MUCH!"                             ID
•71 IFQ<ITHENPRINT"[CLEAR]YOU DIDN'T EAT
 ENOUGH!"                                      ED
•72 FORT=241TO135STEP-1:POKEU,T:NEXT:POKE
 U,0:FORTT=1TO100:NEXT:POKEU,140:FORT=1TO
 1000                                          JJ
•73 NEXT:POKEU,0:PRINT"[DOWN][DOWN]";D;"[
 s V]";E;"=";I;:PRINT"[DOWN][DOWN]YOU ATE
 ";Q                                           AH
•74 K=K-1:PRINT"[DOWN][DOWN]YOUR SCORE IS
 NOW";K:GOTO21                                 OF
•90 POKE52,28:POKE56,28:CLR:FORI=7168TO76
 79:POKEI,PEEK(I+25600):NEXT                   NL
•91 RESTORE:FORT=1TO20:READF:NEXT:FORG=71
 68TO7221:READF:POKEG,F:NEXT                   AO
•92 DATA135,145,155,145,155,165,155,165,1
 75,185,185,195,165,165,165,195,195,165,1
 65,215                                        MK
•93 DATA24,40,122,114,60,24,24,60,0,12,15
 8,253,247,144,12,0,0,48,121,191,239,9,48
 ,0                                            EH
•94 DATA14,7,143,223,255,255,223,143,0,0,
 192,160,240,184,128,240,0,0,0,0,4,28,60,
 28,0,0,0                                      II
•95 DATA0,0,0                                  HD
•100 POKE36879,8:PRINT"[CLEAR][CYAN][3"[c
  +]"][SS][3"[c +]"][SS][3"[c +]"][SS][c
 +][SS][c +]":PRINT"[c +][4"[SS]"][c +][S
 S][SS][c +][3"[SS]"][c +][SS][c +]":PRIN
 T"[c +][c +][3"[SS]"][c +][SS][SS][3"[c
 +]"][SS][3"[c +]"]"                           CG
•101 PRINT"[c +][4"[SS]"][c +][4"[SS]"][c
  +][SS][c +][SS][c +]":PRINT"[c +][3"[SS
 ]"][3"[c +]"][SS][3"[c +]"][SS][c +][SS]
 [c +]":PRINT:PRINT"[3" "][c +][3"[SS]"][
 c +][SS][3"[c +]"][SS][3"[c +]"][SS][c +
```

```
][SS][c +]"                                           GA
·102 PRINT"[3" "][c +][c +][SS][c +][c +]
[SS][c +][SS][c +][SS][SS][c +][SS][SS][
c +][SS][c +]":PRINT"[3"[SS]"][c +][SS][
c +][SS][c +][SS][3"[c +]"][SS][SS][c +]
[SS][SS][3"[c +]"]":PRINT"[3" "][c +][3"
[SS]"][c +][SS][c +][SS][c +][SS][SS][c
+][SS][SS][c +]
 [c +]"                                               BO
·103 PRINT"[3" "][c +][3"[SS]"][c +][SS][
c +][SS][c +][SS][SS][c +][SS][SS][c +][
SS][c +]":PRINT"[DOWN][DOWN][YELLOW][3"
"]BY KEVIN DEWEY."                                    PI
·104 PRINT"[DOWN][DOWN][5" "]HIT ANY KEY"
;:POKE198,0                                           BH
·105 PRINT"[BLACK]":POKE36878,15                      AE
·106 RESTORE:FORT=1TO20:READU:POKE36876,U
:FORTT=1TO100:NEXT:POKE36876,0                        NI
·107 GETA$:IFA$<>""THEN29                             PE
·108 NEXT                                             IA
·109 GOTO106                                          CL
```

FASTNEW

FROM PAGE 33

```
·10 REM --------------------------------              CD
·11 REM            FAST NEW                           LK
·12 REM  DON LEWIS V010585 (REV V123184)              JN
·14 REM --------------------------------              CD
·15 GOTO 1000                                         FC
·16 :                                                 DI
·19 REM DOWNLOAD CODE INTO 1541                       BJ
·20 RESTORE:PRINT CHR$(151)                           PO
·30 FOR I=0 TO 102:READ A:POKE 820+I,A                JP
·35 IF (I/10)-INT(I/10)=0 THEN PRINT "[c
P]";                                                  LM
·40 NEXT I                                            MN
·50 FOR I=0 TO 511:READ A:POKE I+9472,A               PE
·55 IF (I/18)-INT(I/18)=0 THEN PRINT "[c
P]";                                                  MM
·60 NEXT I:PRINT CHR$(30);:RETURN                     CH
·99 :                                                 DI
·100 REM PROMPT AND WAIT FOR RESPONSE                 HJ
·105 PRINT:PRINT "INSERT DISK AND HIT "RT
$;                                                    NN
·110 GOSUB 200:IF A$<>CHR$(13) THEN GOTO
110                                                   BC
·120 RETURN                                           IM
·199 :                                                DI
·200 REM GET CHAR FROM KB, WITH PROMPT                HG
·205 POKE 198,0:POKE 204,0                            HG
·210 GET A$:POKE 207,0:IF A$="" THEN GOTO
 210                                                  CJ
·215 POKE 204,1:PRINT " [LEFT]";:RETURN               JF
.299 :                                                DI
·400 REM DO A DISK JOB, THANKS "INSIDE CO
MMODORE DOS" BY DICK EMMERS, DATAMOST                 JH
·410 TY=0:PRINT#15,"M-W"CHR$(8)CHR$(0)CHR
$(2)CHR$(T)CHR$(S)                                    KH
·420 PRINT#15,"M-W"CHR$(1)CHR$(0)CHR$(1)C
HR$(JB)                                               KH
·430 TY=TY+1                                          OK
·440 PRINT#15,"M-R"CHR$(1)CHR$(0)                     OD
·450 GET#15,E$:IF E$="" THEN E$=CHR$(0)               AH
·460 E=ASC(E$)                                        GE
·470 IF TY=500 THEN GOTO 495                          GI
·480 IF E>127 THEN GOTO 430                           AL
·485 IF E<>1 THEN GOTO 495                            CN
·490 RETURN                                           IM
·495 PRINT:PRINT"FATAL ERROR TRY AGAIN":S
TOP                                                   NN
·499 :                                                DI
·1000 REM ***** MAIN CODE HERE *****                  IC
·1003 Q$=CHR$(5)+"Q"+CHR$(30)                         CK
·1005 POKE 53280,11:POKE 53281,0:PRINT "[
GREEN]";:RT$=CHR$(158)+"<RETURN>"+CHR$(3
0)                                                    PG
·1006 TP$=CHR$(31)+"[s U][19"[s C]"][s I]
"+CHR$(30)                                            DF
·1007 PN$=CHR$(31)+"[s B]"+CHR$(30)+" FAS
T NEW  V010585 "+CHR$(31)+"[s B]"+CHR$(3
0)                                                    EO
·1008 TB$=CHR$(31)+"[s J][19"[s C]"][s K]
"+CHR$(30)                                            CO
·1010 PRINT CHR$(147)CHR$(13)"[10" "]"TP$             GE
·1011 PRINT "[10" "]"PN$                              OC
·1012 PRINT "[10" "]"TB$:PRINT                        KI
·1015 IF FG=1 THEN GOTO 1035                          AF
·1020 PRINT "PLEASE WAIT..";                          JI
·1030 GOSUB 20:PRINT:FG=1                             HG
·1035 SYS 820:DN$="NAME ME"                           AA
·1040 POKE 198,0:INPUT "DISK NAME";DN$                EN
·1050 POKE 198,0:INPUT "DISK ID";DI$                  GL
·1060 DI$=LEFT$(DI$+"ZZ",2)                           CL
·1070 I1=ASC(LEFT$(DI$,1)):I2=ASC(RIGHT$(
DI$,1))                                               JM
·1080 OPEN 15,8,15:PRINT#15,"M-W"CHR$(18)
CHR$(0)CHR$(2)CHR$(I1)CHR$(I2);:CLOSE15               KH
·1090 GOSUB 100                                       CJ
·2000 OPEN 15,8,15:PRINT#15,"M-E"CHR$(0)C
HR$(5);:CLOSE15:OPEN 15,8,15                          DI
·2010 T=18:S=0:JB=176:GOSUB 400:REM SEEK              EA
·2020 JB=128:GOSUB 400:REM READ                       DL
·2030 PRINT#15,"M-W"CHR$(0)CHR$(4)CHR$(3)
CHR$(18)CHR$(1)CHR$(65)                               HG
·2040 JB=144:GOSUB 400:REM WRITE                      IK
·2050 S=1:JB=128:REM READ                             BN
·2060 PRINT#15,"M-W"CHR$(0)CHR$(4)CHR$(2)
CHR$(0)CHR$(255)                                      LC
·2070 JB=144:GOSUB 400:REM WRITE                      IK
·2080 CLOSE 15                                        AB
·2090 OPEN 15,8,15:PRINT#15,"N0:"DN$                  DN
·2100 CLOSE 15                                        AB
·3000 PRINT:PRINT:PRINT"FORMAT COMPLETE,
REMOVE DISK"                                          DM
·3010 PRINT:PRINT"HIT "RT$" TO FORMAT ANO
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering ***Ahoy!*** programs. Refer to these pages **before** entering any programs!

```
 THER DISK,"                                  LM
•3015 PRINT Q$" TO QUIT";:GOSUB 200           PD
•3020 IF A$=CHR$(13) THEN GOTO 1000           BO
•3030 END                                     IC
•4000 DATA 169,16,133,255,169,0,133,251       CK
•4010 DATA 169,37,133,252,169,0,133,253       PE
•4020 DATA 169,5,133,254,165,186,32,177       EL
•4030 DATA 255,169,111,32,147,255,165,251     EM
•4040 DATA 164,252,141,152,3,140,153,3        HC
•4050 DATA 160,0,185,149,3,32,168,255         HM
•4060 DATA 200,192,6,208,245,160,0,177        PF
•4070 DATA 251,32,168,255,200,192,32,144      BF
•4080 DATA 246,165,251,105,31,133,251,165     EH
•4090 DATA 252,105,0,133,252,165,253,105      AG
•4100 DATA 32,133,253,165,254,105,0,133       HH
•4110 DATA 254,32,174,255,198,255,208,180     EP
•4120 DATA 96,77,45,87,0,0,32                 PK
•4999 REM CODE DOWNLOADED INTO 1541 RAM       HI
•5000 DATA 120,173,0,28,9,12,141,0            GO
•5010 DATA 28,169,45,133,74,32,210,6          FM
•5020 DATA 198,74,208,249,162,0,32,217        PM
•5030 DATA 6,169,1,133,34,165,34,201          GG
•5040 DATA 36,240,12,32,70,5,48,7             MM
•5050 DATA 230,34,32,200,6,240,238,173        HN
•5060 DATA 0,28,41,243,141,0,28,169           ED
•5070 DATA 236,141,12,28,165,75,240,5         LB
•5080 DATA 41,127,76,200,193,96,32,75         IF
•5090 DATA 242,133,67,138,10,10,10,10         PO
•5100 DATA 10,133,68,173,0,28,41,159          EO
•5110 DATA 5,68,141,0,28,169,238,141          GN
•5120 DATA 12,28,160,0,132,77,165,57          GJ
•5130 DATA 153,0,3,200,200,165,77,153         OD
•5140 DATA 0,3,200,165,34,153,0,3             EM
•5150 DATA 200,165,19,153,0,3,200,165         FG
•5160 DATA 18,153,0,3,200,169,15,153          GP
•5170 DATA 0,3,200,153,0,3,200,185            ID
•5180 DATA 250,2,89,251,2,89,252,2            EO
•5190 DATA 89,253,2,153,249,2,230,77          KN
•5200 DATA 165,77,197,67,144,192,152,72       GB
•5210 DATA 169,3,133,49,32,48,254,104         II
•5220 DATA 168,136,32,229,253,32,245,253      PM
•5230 DATA 169,1,162,0,157,0,4,232            HG
•5240 DATA 208,250,169,4,133,49,32,233        DK
•5250 DATA 245,133,58,32,143,247,169,206      HM
•5260 DATA 141,12,28,169,255,141,1,28         KM
•5270 DATA 141,3,28,32,243,6,169,0            OD
•5280 DATA 133,50,169,255,141,1,28,162        AM
•5290 DATA 5,80,254,184,202,208,250,162       BG
•5300 DATA 10,164,50,80,254,184,185,0         ED
•5310 DATA 3,141,1,28,200,202,208,243         NN
•5320 DATA 162,9,80,254,184,169,85,141        ML
•5330 DATA 1,28,202,208,245,162,5,169         KA
•5340 DATA 255,80,254,184,141,1,28,202        NH
•5350 DATA 208,247,162,187,80,254,184,189     PI
•5360 DATA 0,1,141,1,28,232,208,244           GG
•5370 DATA 160,0,80,254,184,177,48,141        PG
•5380 DATA 1,28,200,208,245,169,85,162        HI
•5390 DATA 8,80,254,184,141,1,28,202          JK
•5400 DATA 208,247,165,50,24,105,10,133       NP
•5410 DATA 50,198,77,208,149,80,254,184       IE
•5420 DATA 80,254,184,32,0,254,169,0          PB
•5430 DATA 133,48,169,3,133,49,165,67         LJ
•5440 DATA 133,77,32,176,6,162,10,80          ML
•5450 DATA 254,184,173,1,28,209,48,208        DB
•5460 DATA 14,200,202,208,242,24,165,48       II
•5470 DATA 105,10,133,48,76,132,6,169         BF
•5480 DATA 244,133,75,96,32,176,6,160         OA
•5490 DATA 187,80,254,184,173,1,28,217        JI
•5500 DATA 0,1,208,235,200,208,242,162        AE
•5510 DATA 252,80,254,184,173,1,28,217        CD
•5520 DATA 0,4,208,219,200,202,208,241        ML
•5530 DATA 198,77,208,182,169,0,240,209       ML
•5540 DATA 169,208,141,5,24,169,161,44        MN
•5550 DATA 5,24,16,197,44,0,28,48             FL
•5560 DATA 246,173,1,28,184,160,0,96          HK
•5570 DATA 32,203,6,174,0,28,232,76           LH
•5580 DATA 217,6,32,213,6,174,0,28            EL
•5590 DATA 202,138,41,3,133,68,173,0          BP
•5600 DATA 28,41,252,5,68,141,0,28            EL
•5610 DATA 160,8,162,0,202,208,253,136        OI
•5620 DATA 208,250,96,169,85,141,1,28         CK
•5630 DATA 160,32,208,240,234,234,234,234     PK
•60000 CLOSE15:OPEN 15,8,15:INPUT#15,A$,B
 $,C$,D$                                      LK
•60001 PRINT A$"[SS]"B$"[SS]"C$"[SS]"D$:C
 LOSE15:STOP                                  LM
•61000 CLOSE15:OPEN 15,8,15,"I0:":SAVE "@
 0:FASTNEW.BAS",8:CLOSE 15:STOP               KL
•62000 CLOSE 15:OPEN 15,8,15:PRINT#15,"M-
 R"CHR$(18)CHR$(0):GET#15,I1$                 DP
•62001 PRINT#15,"M-R"CHR$(19)CHR$(0):GET#
 15,I2$                                       EO
•62002 PRINT I1$I2$                           NF
•62004 CLOSE 15:STOP                          NG
```

INVECTIVE
FROM PAGE 15

FROM PAGE 15

```
•10 REM PROGRAM: INVECTIVE                    JC
•20 REM GEORGE TREPAL                         BM
•30 REM 2650 ALTURAS RD                       PO
•40 REM BARTOW, FL  33830                     AG
•50 :                                         DI
•100 GOSUB 1150                               FK
•110 POKE53280,12:POKE53281,12:PRINT"[CLE
 AR]";                                        KE
•120 REM SET UP SCREEN                        AL
•130 G$(1)="[BLACK][c B]  [YELLOW][6"[s Z
 ]"][3" "][6"[s Z]"][4" "][6"[s Z]"][3" "
 ][6"[s Z]"]  [BLACK][c B]                    KN
•140 G$(2)="[BLACK][c B]  [BLUE][c +][4"
 "][c +][3" "][c +][4" "][c +][4" "][c +]
```

```
[4" "][c +][3" "][c +][4" "][c +]  [BLAC
K][c B]"                                 LC
·150 G$(3)="[BLACK][40"[c B]"]"          IB
·160 G$(4)="[BLACK][c B][4" "][YELLOW][s
Z][8" "][s Z][10" "][s Z][8" "][s Z][4"
"][BLACK][c B]"                          LE
·170 PRINT G$(3); G$(4); G$(2); G$(2); G$
(1); G$(2); G$(2); G$(4);                BF
·180 PRINT G$(4); G$(4); G$(2); G$(2); G$
(1); G$(2); G$(2); G$(4);                DC
·190 PRINT G$(4); G$(4); G$(2); G$(2); G$
(1); G$(4); G$(4); G$(3);                CF
·200 :                                   DI
·210 REM POKE GREEN POINTS               EG
·220 CM=54272:  REM CM=COLOR MEMORY      KB
·230 FOR J=1 TO 50                       DM
·240 RN = INT(RND(1)*920)+1024           PB
·250 IF PEEK(RN)<>32 THEN 240            LB
·260 POKE RN,105: POKE RN+CM,13          KG
·270 NEXT                                IA
·280 :                                   DI
·290 REM SET JOYSTICK VALUES J() AND SCRA
MBLE ARRAY VALUES SA()                   FK
·300 J(1)=-40: J(2)=40: J(4)=-1: J(8)=1  CO
·310 SA(1)=1: SA(2)=2: SA(3)=4: SA(4)=8  CA
·320 :                                   DI
·330 REM SET UP PLAYER  SP=SPACES OVER  D
N$=HOW FAR DOWN  P=POSITION              CI
·340 REM BM$=BOTTOM OF SCREEN  TI$=TIMER NF
·350 PRINT"[HOME][DOWN][WHITE][RIGHT][s Q
]"                                       PL
·360 SP=1: DN$="[WHITE][HOME][DOWN]": P=1
065: TI$="[5"0"]1"                       IL
·370 BM$="[HOME][24"[DOWN]"]": GOSUB 750:
 GOSUB 1000                              MM
·380 :                                   DI
·390 REM READ PORT 2 JOYSTICK            PA
·400 JV=PEEK(56320)                      GA
·410 JV=15-(JVAND15)                     GH
·420 PRINTBM$;"[WHITE]SCORE ";SC;        EL
·430 VT=VAL(TI$): IF VT/15=INT(VT/15)THEN
 GOSUB 680                               IO
·440 IF JV=0 THEN 400                    LA
·450 :                                   DI
·460 REM FIND CHARACTER MOVED TO         MB
·470 T=P+J(JV)                           ED
·480 CH=(PEEK(T))                        AB
·490 IF CH=32 THEN GOSUB 570:  GOTO 400:
REM BLANK SPACE                          CF
·500 IF CH=105 THEN SC=SC+10: GOSUB 570:
GOSUB 1000: GOTO 400: REM GREEN SCORE    JH
·510 REM BLUE JOYSTICK SCRAMBLE          PM
·520 IF CH=102 THEN GOSUB 570: GOSUB 680:
 SC=SC-20: GOTO400                       EL
·530 IF CH=90 THEN SC=SC-50: GOSUB 570: G
OSUB 950: GOTO 400: REM YELLOW SCORE     DP
·540 GOTO400                             CA
·550 :                                   DI
·560 REM WIPE OUT PLAYER POSITION        NI
·570 PRINT DN$; SPC(SP)" ";              LL
·580 :                                   DI
·590 REM PRINT NEW PLAYER POSITION       OD
·600 P=T                                 FB
·610 IF J(JV)=1 THEN SP=SP+1             MB
·620 IF J(JV)=-1 THEN SP=SP-1            FB
·630 IF J(JV)=40 THEN DN$=DN$ + "[DOWN]" AK
·640 IF J(JV)=-40 THEN DN$=LEFT$(DN$,LEN(
DN$)-1)                                  LE
·650 PRINT DN$; SPC(SP)"[s Q]";: RETURN  ID
·660 :                                   DI
·670 REM SCRAMBLE JOYSTICK               LB
·680 IF VT>130 THEN 1050                 ED
·690 FOR J=1 TO 4: RN=INT(RND(1)*4)+1    GG
·700 T=SA(J): SA(J)=SA(RN): SA(RN)=T: NEX
T                                        GH
·710 J(SA(1))=1: J(SA(2))=-1: J(SA(3))=40
: J(SA(4))=-40                           FF
·720 GOSUB 900                           EB
·730 :                                   DI
·740 REM POSITION HINTS                  LH
·750 PRINT BM$;SPC(13)"[c 2]";           AJ
·760 FOR J=1 TO 8                        AL
·770 IF J=1 THEN PRINT "UP ";            LN
·780 IF J=2 THEN PRINT "DN ";            JD
·790 IF J=4 THEN PRINT "LT ";            LL
·800 IF J=8 THEN PRINT "RT ";            KF
·810 IF J(J)=1 THEN PRINT ">  ";         PG
·820 IF J(J)=-1 THEN PRINT "<  ";        FN
·830 IF J(J)=40 THEN PRINT "V[3" "]";    DP
·840 IF J(J)=-40 THEN PRINT "[UPARROW]  "
;                                        PF
·850 NEXT                                IA
·860 POKE 53280,2: POKE 53281,2: FOR DL=
1 TO 80: NEXT                            EG
·870 POKE 53280,12: POKE 53281,12        NO
·880 :                                   DI
·890 REM SCRAMBLE SOUND                  KE
·900 POKE 54296,15: POKE 54276,17: POKE 5
4277,15: POKE 54273,60                   HP
·910 FOR DL=1 TO 25:NEXT                 BG
·920 POKE 54296,0: POKE 54276,0: RETURN  EB
·930 :                                   DI
·940 REM YELLOW HIT                      KD
·950 POKE 54296,15: POKE 54276,33: POKE 5
4277,15: POKE 54273,40                   KL
·960 FOR DL=1 TO 25:NEXT                 BG
·970 POKE 54296,0: POKE 54276,0: RETURN  EB
·980 :                                   DI
·990 REM GREEN SOUND                     LN
·1000 POKE 54296,15: POKE 54276,17: POKE
54277,15: POKE 54273,30                  KA
·1010 FOR DL=1 TO 25:NEXT                BG
·1020 POKE 54296,0: POKE 54276,0: RETURN EB
·1030 :                                  DI
·1040 REM END GAME                       OE
·1050 PRINT"[CLEAR][8"[DOWN]"]";SPC(15);"
```

```
 SCORE ";SC                                OF
•1060 FOR J=1 TO 4: GOSUB 900: GOSUB 950:
 GOSUB 900                                 PI
•1070 GOSUB 1000:NEXT                      DI
•1080 PRINT"[4"[DOWN]"]  PRESS JOYSTICK B
UTTON TO PLAY AGAIN"                       BF
•1090 JV=PEEK(56320)                       GA
•1100 FR=JVAND16                           AN
•1110 IF FR = 16 THEN 1090                 ED
•1120 GOTO 110                             DM
•1130 :                                    DI
•1140 REM INTRO                            DH
•1150 PRINT"[CYAN][CLEAR][DOWN]":POKE 532
80,0: POKE 53281,0:                        CA
•1160 PRINTSPC(16)"INVECTIVE[DOWN]"        NI
•1170 PRINTSPC(12)"BY GEORGE TREPAL[DOWN]
[DOWN]"                                    DA
•1180 PRINTSPC(11)"[c 6]GREEN = 10 POINTS
"                                          PH
•1190 PRINTSPC(10)"[DOWN][YELLOW]YELLOW =
 -50 POINTS"                               IP
•1200 PRINTSPC(2)"[DOWN][c 7]BLUE = -20 P
OINTS AND SCRAMBLES THE"                   JP
•1210 PRINTSPC(16)"JOYSTICK[DOWN]"         IA
•1220 PRINTSPC(2)"[CYAN]THE JOYSTICK IS S
CRAMBLED EVERY 15"                         NN
•1230 PRINTSPC(2)"SECONDS WHETHER YOU HIT
 BLUE OR NOT"                              CD
•1240 PRINTSPC(5)"[DOWN]THE JOYSTICK GOES
 IN PORT #2"                               IB
•1250 PRINTSPC(7)"[DOWN]THE GAME LASTS 90
 SECONDS"                                  AC
•1260 PRINTSPC(5)"[DOWN]PRESS THE FIRE BU
TTON TO PLAY"                              EJ
•1270 JV=PEEK(56320)                       GA
•1280 FR=JVAND16                           AN
•1290 IF FR = 16 THEN 1270                 EH
•1300 RETURN                               IM
```


...COMING IN THE OCTOBER *AHOY!* (ON SALE SEPTEMBER 3)...

ALIVE AND IN LIVING COLOR
TURN YOUR C-64 SPRITES INTO FULL COLOR ANIMATED CHARACTERS

INFINITESIMAL INTRIGUE
COMPUTER EXPLORATIONS OF SUB-MICROSCOPIC ACTIVITY

PROGRAMMING BIT BY BIT
PERFORMING SINGLE BIT MANIPULATIONS ON BINARY NUMBERS

DON'T MISS THE BOAT!



SUBSCRIBE TO AHOY!

- ☐ Send me twelve issues of AHOY! for $19.95 ($26.95 Canada & elsewhere). I understand that I will be billed later.
- ☐ Send me twenty-four issues of AHOY! for $37.95 ($49.95 Canada & elsewhere). I understand that I will be billed later.

NAME ______________________

ADDRESS ______________________

CITY ______________________

STATE __________ **ZIP** __________

READER SERVICE CARD

To request additional information on any product advertised or otherwise pictured in this issue of *Ahoy!* that is accompanied by a reader service number, circle the corresponding number below and mail this card. We will promptly forward your request to the designated companies.

101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300

PLEASE ANSWER THE QUESTIONS BELOW.

Sex and age
A ☐ male B ☐ female age________

Educational level completed
C ☐ elementary
D ☐ high school
E ☐ junior college
F ☐ college graduate
G ☐ masters degree
H ☐ phd

From which of the following sources did you obtain your copy?
K ☐ newsstand N ☐ subscription (mail)
L ☐ from a friend or family member
M ☐ other ____________________

If not currently a subscriber, do you plan to become one?
I ☐ yes J ☐ no

Name ______________________

Street ______________________

City __________ **State** __________ **Zip** __________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 65 MT. MORRIS, ILL.

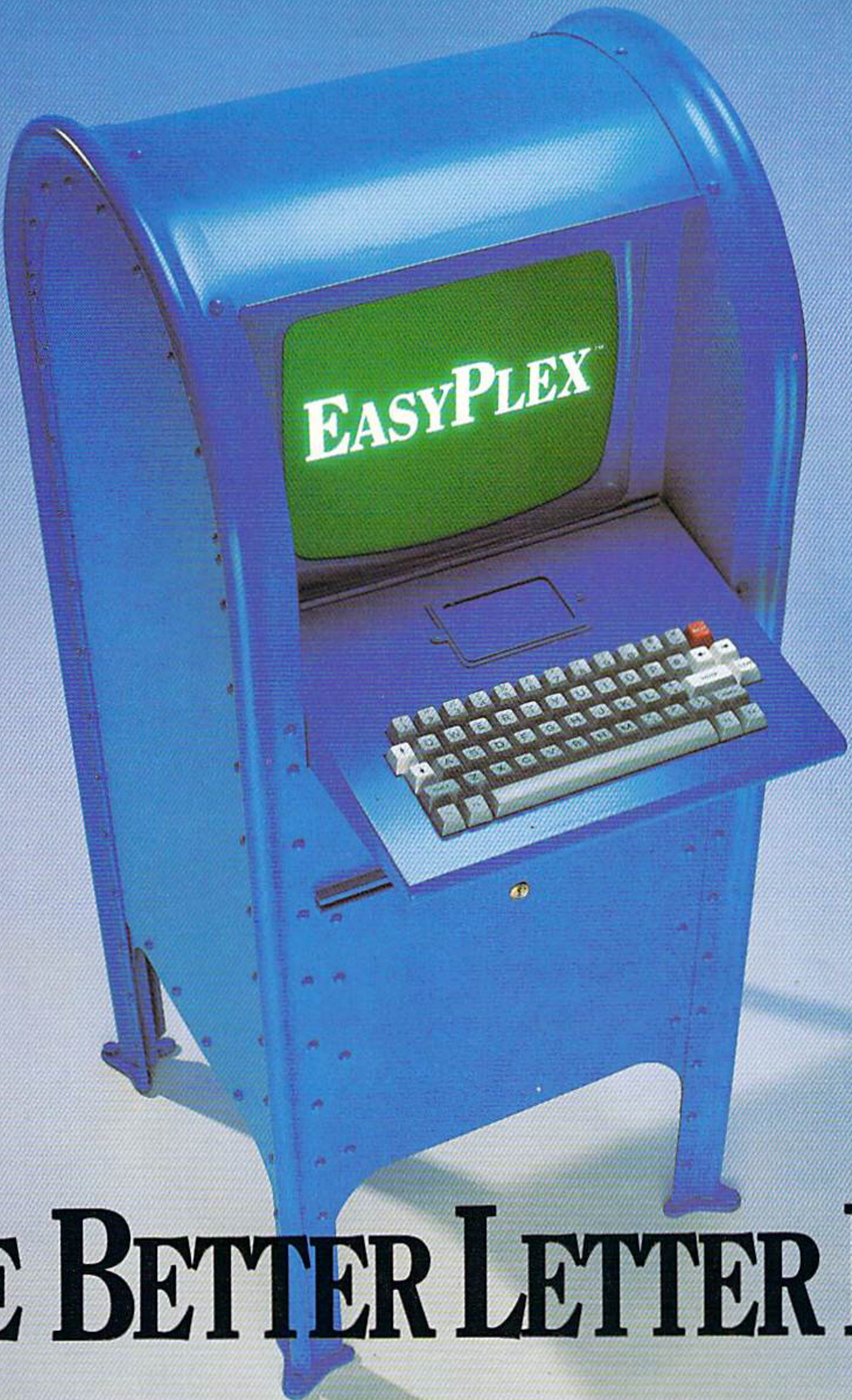
POSTAGE WILL BE PAID BY ADDRESSEE

Ahoy!

P.O. BOX 341
MT. MORRIS, ILL. 61054

PLACE
STAMP
HERE

AHOY! MAGAZINE
P.O. Box 2501
Clinton, Iowa 52735



THE BETTER LETTER BOX

Introducing EasyPlex™. The new, easy-to-use electronic mail system from CompuServe.

Finally! Electronic Mail that's so easy to use you can start composing and sending messages the first time you get online.

Designed for various experience levels, EasyPlex has a menu mode with simple, easy-to-follow directions for beginners, and it lets experienced users save time by working in the prompt or command modes. With EasyPlex, you can compose, edit, send, file, and take advantage of sophisticated options previously available only with more expensive services.

CompuServe's EasyPlex lets friends and relatives, associations and club members communicate any time of the day or night. And small business owners, real estate professionals, insurance agents, lawyers, writers, etc. can communicate quickly and simply— either interoffice or interstate. "It's Easy." "Just Plex it!"

Best of all, EasyPlex is available to all CompuServe subscribers. And, along with EasyPlex, you get hundreds of valuable and entertaining computing options. Plus the assurance of belonging to the largest, fastest growing computer information service in the world and the premier supplier of business information to FORTUNE 500 companies.

Start communicating! To buy a CompuServe Subscription Kit, see your nearest computer dealer. To receive our informative brochure or to order direct, call or write:

CompuServe®

Information Services, P.O. Box 20212
5000 Arlington Centre Blvd., Columbus, Ohio 43220

800-848-8199

In Ohio, call 614-457-0802

An H&R Block Company

Reader Service No. 144